



Escuela
Politécnica
Superior

Proyecto Argos:

Sistema de monitorización de personas en riesgo

Grado en Ingeniería Informática



Trabajo Fin de Grado

Autor:

Jaime Sarrión Sahuquillo

Tutor/es:

Otto Colomina

Septiembre 2019



Universitat d'Alacant
Universidad de Alicante

AGRADECIMIENTOS

A mi compañero de trabajo y gran apasionado del IoT que me dio la idea para el TFG

Rubén Hernández Vicente

Por el soporte que me han brindado,

A mis padres

Por haberme dejado llevar a cabo la loca idea que le presenté,

A mi tutor Otto Colomina

Por encenderme la bombilla de las ideas cuando se me apagaba,

A Lawrence Rider Arthur García y Jaume Moreno Cantó

Por ayudarme con el diseño y la edición de Argos,

A Eva María Saravia González

INDICE

Contenido

AGRADECIMIENTOS	2
INDICE	3
INTRODUCCIÓN	5
ESTADO DEL ARTE	6
Dispositivos IoT	6
Luces inteligentes	6
Arduino	7
Asistentes	7
Dispositivos de monitorización	10
Holter	11
Proyecto Lucia	12
Dispositivos LoRa	12
Semtech SX1261	13
Semtech LLCC68	13
Problemática de los Hospitales y centros de salud	13
METODOLOGIAS DE DESARROLLO	15
HERRAMIENTAS Y TECNOLOGIAS	18
Raspberry Pi	18
TTGO LoRa32 esp32 OLED	18
Adobe Photoshop	20
Trello	20
Git	21
Arduino IDE	22
VSCode	23
React	24
React-Native	24
NodeJS	25
MySQL	26
CSS	26
ARQUITECTURA DEL SISTEMA	28
Modulo LoRa Cliente	29
Modulo LoRa Servidor	29

Servidor API REST	30
Cliente móvil	30
PROCESO DE DESARROLLO	34
Búsqueda de información	34
¿Qué es LoRa y sus características?	34
¿Qué es LoRa WAN?	35
Clases de dispositivos	35
Elementos necesarios para el desarrollo del proyecto	36
Hardware	36
Hola mundo en LoRa	36
Transmitir un mensaje entre placas	39
Como conectar una de las placas con la Raspberry	40
Montaje de la API en la Raspberry	42
Conexión del sensor de ritmo cardiaco	43
Software	43
Creación de los mockups	43
Diseño de Argos	44
Diseño de BD	46
Diseño de la API	47
BUSINES Y FUTURO	49
CONCLUSIONES	50
REFERENCIAS Y BIBLIOGRAFIA	51

INTRODUCCIÓN

Este trabajo de final de carrera hace uso de LoRa, una nueva tecnología de comunicación que más tarde se describirá en profundidad en el apartado correspondiente. Además, se pretende hacer ver los diversos usos que se le pueden dar en el campo del IoT. Se describirán los pasos seguidos para crear una pequeña arquitectura para el cuidado de personas en riesgo.

El objetivo a corto plazo de este proyecto, es hacer ver que, con un presupuesto limitado y algunas nociones de programación, se puede hacer un sistema bastante fiable para la monitorización de personas que padecen alguna enfermedad y hay que monitorizar las constantes vitales, así como, las pulsaciones, el oxígeno en sangre o los pasos que ha realizado en el día de hoy. Además, todo esto vendrá acompañado con una aplicación para el móvil y un API sencilla.

El objetivo a largo plazo, sería la puesta en marcha de este proyecto a mayor escala, para la monitorización de personas en riesgo. En un primer lugar, lo que podríamos hacer es instalar unas antenas en los centros de salud de las ciudades, lo cual, nos permitiría tener un radio de unos 10 km² cubiertos. Además, podríamos perfeccionar el diseño del sistema que monitoriza nuestras constantes, para que sea una simple pulsera que podamos llevar en la muñeca cómodamente. En cuanto a nivel de software, hablamos de la creación de un sistema donde sea mucho más fácil la comunicación médico-usuario y la optimización de las consultas.

Al ser un proyecto al que le he dedicado mucho tiempo, decidí ponerle nombre y con él una personalidad. Por ello, decidí llamarlo Argos. Argos, en la mitología griega era un gigante encargado de custodiar a Ío, una de las amantes de Zeus. Gracias a los mil ojos que tenía el gigante era un guardián excepcional, por ello, como esta aplicación es un sistema de monitorización de personas en riesgo y al fin y al cabo también es algo así como un guardián con mil ojos, he decidido llamarla Argos.

ESTADO DEL ARTE

Me dispongo a mostrar y desglosar, los diferentes dispositivos más utilizados que hay hoy en día en el mundo del IoT, ya que, el hardware del que voy a hacer uso en este proyecto, no sólo lo podemos usar para monitorizar a gente en riesgo, sino que también podríamos darle alguna aplicación en el campo de la domótica. Por ello, me dispongo a proporcionar la información de cómo está ahora mismo el mundo de la domótica y de la monitorización de personas.

Además, en pleno siglo XXI, con el auge de las tecnologías podemos observar que en ciertos sectores como en el de la salud, no se está invirtiendo lo suficiente en modernizar los sistemas que tienen tanto los hospitales como los centros de salud

Además, me dispongo a mostrar los problemas a los que se enfrentan el personal sanitario y la necesidad que hay de unificar un sistema único para todos los hospitales y ambulatorios.

Dispositivos IoT

Hoy en día tenemos infinidad de dispositivos IoT en el mercado, los cuales podemos comprar a un precio razonable y pueden hacer nuestra vida mucho más sencilla. La mayoría de estos dispositivos ya vienen preparados para su uso, siguiendo con la filosofía 'plug and play', aunque todavía existen algunos que requieren un poco más de trabajo para funcionar. A continuación, me dispongo a detallar algunos de ellos.

Luces inteligentes

La búsqueda de la automatización de las tareas más simples de la casa ha llegado incluso a la iluminación. Por ello, ha surgido lo que se denomina como iluminación inteligente. Con un sistema automatizado, podemos mejorar la iluminación en distintas partes de la casa, así como, mejorar el rendimiento, la eficacia y por qué no, la seguridad.

La instalación de estos dispositivos lumínicos puede variar dependiendo de lo que se busque y va desde un cableado propio de control, a través de la propia red eléctrica o de forma inalámbrica. La mayoría de estos dispositivos se instalan sin la necesidad de que haya una instalación domótica previa. Además, la mayoría de marcas nos brindan aplicaciones móviles, con las que controlar o simplemente ver algunos detalles como el consumo o similares.

Algunos ejemplos pueden ser: los dispositivos [Philips](#) que oscilan entre los 51 a los 86€ por unidad; algo más económico, son los dispositivos de [Xiaomi](#) que van desde los 20€

hasta los 100€ lo cual nos brinda una alta gama de productos.



Arduino

Por supuesto cuando hablamos de dispositivos IoT no podemos dejarnos a Arduino. Cuando hablamos de Arduino, no podemos evitar pensar en esas pequeñas placas que alguna vez en nuestra vida hemos visto, pero lo cierto es que va mucho más allá. Arduino es una plataforma 'open-source' que nos brinda la posibilidad de obtener tanto su hardware como su software libre, flexible y realmente fácil de usar.

El hardware de Arduino es una placa con un microcontrolador ATMEL, en este, podemos insertar ordenes o instrucciones a través del IDE de Arduino. Estas órdenes nos permiten crear programas e incluso interactuar con otros dispositivos. Lo bueno de este sistema, es que es barato y relativamente fácil de usar gracias a su IDE, es por ello, que cada vez más gente lo usa para sus dispositivos IoT.

Asistentes

Gracias a los avances en reconocimiento de voz, machine learning, etc, podemos disfrutar hoy en día de lo que denominan asistentes virtuales. Estos asistentes son nada más que un software que a través de comandos de voz, son capaces de realizar las tareas que les ordenamos.

Por otro lado, ahora no sólo los podemos disfrutar en el Smartphone, sino que también han dado el salto al mundo del IoT, y te permiten automatizar tareas de la casa (encender la tele, las luces, activar o desactivar enchufes, etc.). Algunas compañías incluso han creado aplicaciones donde te aparecen todos los dispositivos IoT que están conectados a tu red y te permiten configurarlos a tu gusto. Por ejemplo, en el caso de Google Home, te permite configurar tu cuenta de Spotify para que cuando le pidas una canción, por defecto la abra en esa plataforma. Esta amplia variedad de cosas que se pueden hacer, junto con su bajo coste, hace que sea mucho más fácil tener una casa inteligente.

Siri



Siri comenzó siendo un proyecto independiente, pero no fue hasta que Apple decidió introducirla en sus Iphone 4S cuando realmente dio el salto a la fama. Siri es una inteligencia artificial que hace uso de funciones de procesamiento de lenguaje natural para realizar tareas, responder preguntas, etc.

En un principio Siri sólo podía encargarse de unas tareas básicas que iban desde acceder a una página hasta reservar en un restaurante. Actualmente, Siri es capaz de escribir o leer mensajes de las redes sociales, incluso de organizar tu agenda. Como contrapartida, diremos que Siri sólo está disponible en dispositivos Apple, que no cuenta con la capacidad de aprender de nuestros gustos personales y que sólo permite conversación por voz.

Google Now



Google Now es el sistema desarrollado por Google, el cual entró en el mercado dentro de la aplicación Google Search para los sistemas operativos Android 4.1 en el 2012. Google también se suma a los procesadores de lenguaje natural con este Google Now, el cual, es capaz de responder preguntas, hacer recomendaciones, etc. Actualmente, Google Now está disponible para otros sistemas operativos como IOS, accesible a través de google Chrome.

En cuanto a las ventajas y desventajas de este asistente hay varias cosas a tener en cuenta, como por ejemplo que es capaz de personalizar las respuestas en base a tu historial, además cuenta con un sistema de ayuda pasivo, donde antes de realizar

cualquier acción, te muestra las posibles búsquedas o acciones que vas a realizar, en base a la experiencia. El lado malo de este asistente, es que muchos de los usuarios se han quejado de la cantidad de datos personales que Google almacena, lo cual deja en tela de juicio nuestra privacidad.



Cortana



Este es el asistente desarrollado por Microsoft, el cual viene integrado en Windows 10 de manera inseparable. Pese a ser de Windows, también podemos disfrutar de sus servicios a través de una app independiente e incluso en un futuro, veremos este asistente a bordo de los coches de los grupos Nissan y BMW. Este asistente también hace uso de las funciones de reconocimiento de lenguaje natural, aunque también es posible transmitirle las ordenes a través del teclado.

En términos generales Cortana se parece bastante a otros asistentes como Siri, a pesar de que presenta algunas ventajas respecto a este. Una de las características más destacadas es Cortana recoge tus datos, personaliza tu búsqueda y te ofrece consejos y sugerencias. Un punto en contra, sería que las sincronizaciones con otras aplicaciones son limitadas en comparación a otros asistentes.

Alexa



amazon alexa

Amazon también ha querido meterse en el mundo de los asistentes virtuales, y como no, ha desarrollado uno propio, Alexa. En este caso no había antes precedentes de Alexa, es decir, no se introdujo en ningún dispositivo anteriormente, sino que Amazon lo ha desplegado directamente en formato físico. A pesar de esto, Huawei está estudiando introducirlo en sus terminales.

El dispositivo elegido por la gran compañía de ventas por internet ha sido un altavoz de forma cilíndrica, denominado Amazon Echo. Este es capaz de funcionar sin necesidad de una configuración previa, sólo es necesario decir Alexa, su nombre, para activarlo.



Dispositivos de monitorización

El mundo del IoT también ha llegado al ámbito de la salud, prueba de ello, es que ahora hay infinidad de servicios que son capaces de monitorizar a una persona 24/7, ya no es necesario que esa persona esté en el hospital para poder comprobar sus constantes vitales. Esto es muy útil, ya que a veces es necesario medir esas constantes vitales en la vida cotidiana del individuo, cosa que no puede hacer tumbado en una cama en el

hospital. Es por ello, que la tecnología y en concreto el mundo del IoT ha puesto algunas soluciones a este problema tan común. Me dispongo a mostrar y detallar algunas de ellas.

Copcar

Copcar es un sistema de monitorización en tiempo real que lo que trata es reducir el tiempo de respuesta ante un problema cardiovascular, este software está integrado con los sistemas de emergencia. El funcionamiento de la aplicación es bastante simple, la actividad cardiaca es recogida a través de una serie de sensores, que se lo transmiten a la aplicación móvil, la cual envía las señales a la API, y esta analiza si el paciente está en riesgo o no. En caso de que esté en riesgo, se da parte al 112 y se le mandan las coordenadas de la víctima. Este sistema es similar al que estamos implementando, pero lo cierto es que lo que buscamos es abaratar los costes, además de ofrecer más servicios como la monitorización de los pasos, oxígeno en sangre, etc. El coste del servicio Copcar es de unos 90€/mes que es muchísimo más de lo que me he gastado en todo el montaje del proyecto, y eso, cada mes.



Holter

El Holter o Holter-ecg es un aparato médico que es capaz de registrar la actividad eléctrica del corazón durante varias horas. La ventaja más característica de este aparato es que es capaz de registrar la actividad cardiaca de una persona durante un periodo prolongado, a diferencia del electrocardiograma convencional, el cual sólo puede medir la actividad eléctrica de una persona en un periodo breve de tiempo.

En este caso, el Holter tiene que ser recetado por un médico, y sólo en caso de que haya indicios de que sea necesario, como, por ejemplo, en el caso de que se hayan producido desmayos, palpitaciones, infartos, etc. En esos casos, el médico te recetará un 'Holter ambulatorio', el cual tendrá una duración de unas 24 horas más o menos. Una

vez haya finalizado la prueba deberás entregar el Holter para una posterior evaluación médica.

Esta prueba presenta algunos inconvenientes, por ejemplo, en caso de que le ocurra algo al paciente durante la prueba, sí, quedará registrado, pero no se avisará a emergencias, ya que este dispositivo no está conectado a la red. Otro de los inconvenientes que presenta, es que los datos tienen que ser leídos por personal cualificado, ya que cualquier médico no sabría descifrar con exactitud la información que los datos recogidos le está dando.

Proyecto Lucia

Este proyecto tiene su origen en Tenerife, y lo que busca es crear un hogar inteligente para las personas de la 3ª edad. Dicho hogar tendría una IA que estaría en continuo aprendizaje. La IA intentaría aprender las pautas de conducta de los ancianos y así cuando se produzca una conducta anómala, avisar a las autoridades pertinentes.

La finalidad de este proyecto es brindar una ayuda proactiva a las personas a cargo de los ancianos, para que en caso de que esté ocurriendo algo peligroso para el anciano, como una caída, un infarto o algo similar, se pueda acortar el tiempo de respuesta del personal sanitario y así tener más probabilidades de éxito.

Los desarrolladores de este software han dividido en patrones de comportamiento basándose en la inactividad, la localización y la activación de otros sensores. Algunos de los patrones que describen son: Inactividad normal, descanso, dormido, inactividad anómala, etc. Con estos patrones simples, se quiere llegar a unos más complejos, con los registrar los hábitos de vida de estas personas, por ejemplo, cuantas veces van al baño, las horas de sueño, etc. Todo esto en conjunto podría ayudar incluso a una detección temprana de alguna anomalía en el paciente.

Dispositivos LoRa

Hoy en día resulta realmente sencillo pasarse a la comunicación LoRa, en el mundo IoT, lo único que necesitamos es un chip transceptor, es decir emisor y receptor. Una vez tenemos el chip, podemos usarlo con Arduino, Raspberry o incluso nuestras propias placas si tenemos conocimientos suficientes. Para el proyecto, se ha escogido una solución más simple en la cual viene montado el chip transceptor en una placa ESP32, con bluetooth, wifi y una pantalla OLED (más adelante se describirá con mas detalle).

Semtech SX1261



Este dispositivo es ideal para aplicaciones de largo alcance, como puede ser LoRa. Este dispositivo, está diseñado para una consumir muy poca batería, con solo 4.2 mA de consumo cuando está activo. Este chip puede emitir en frecuencias de hasta +15dBm. Este tipo de dispositivo es altamente configurable para cumplir con diferentes requisitos de aplicación usando el estándar LoRaWAN o protocolos similares.

Semtech LLCC68



El LLCC68 es un transceptor, que se utiliza para comunicaciones de rango medio en interiores y de interiores a exteriores. Este chip presenta las velocidades más altas de LoRaWAN. Este chip, como el anterior, está diseñado para una larga vida útil de la batería, con un consumo de 4.2 mA en estado activo. A diferencia del anterior, este es capaz de transmitir hasta +22dBm con amplificadores de potencia.

Problemática de los Hospitales y centros de salud

Después de haber preguntado a diversos profesionales de la salud e incluso haber estado trabajando para un hospital, desarrollando el software, he podido observar algunas cosas que se deberían corregir. Tanto los empleados de la salud, como los desarrolladores que crean el software, llegan a la misma conclusión cuando hablan del software que se usa en los hospitales, es un caos. Y no es de extrañar, ya que, en un hospital trabajan miles y miles de empleados, cada uno con unos permisos determinados y que desempeña una función concreta dentro del hospital. Por otro lado, tenemos a los pacientes, los cuales se cuentan por cientos de miles, con la cantidad de datos que ello conlleva. Es por todo esto, que urge la necesidad de crear un software, que gestione y facilite el intercambio de comunicación entre todos los que se encuentran en el hospital, tanto pacientes como médicos. Este pensamiento, ya lo han llevado a cabo muchos

hospitales y se han puesto manos a la obra, con lo que ha surgido un nuevo problema, la cantidad de programas diferentes que hay para la gestión de hospitales. Esto a su vez deriva en más problemas, ya que, el intercambio de datos de pacientes entre hospitales es una tarea bastante difícil de llevar a cabo. Además, por si no fueran pocos los problemas, los profesionales de la salud, que llevan años y años formándose, tienen que aprender a usar el programa del hospital, con la particularidad de que, si se cambian de hospital, todo lo aprendido, no vale nada. Por todo esto y mucho más, parece necesaria la creación de un software común para todos los hospitales, el cual permita la interoperabilidad entre estos. Países como Chile, ya se han puesto manos a la obra con el programa SIDRA (Sistema de integración de redes asistenciales).

METODOLOGIAS DE DESARROLLO

Ya hace algunos años que las metodologías ágiles aparecieron y parece que las empresas están empezando a tomar consciencia de lo que les pueden aportar, es por ello, que cada vez son más las que deciden pasarse a este modelo de producir software. Por ello, en la medida de lo posible, se ha intentado emular los pasos que se seguirían en un desarrollo ágil de un pequeño grupo.

Lo primero que se llevaría a cabo, sería la implementación de las historias de usuario. Dichas historias las he llevado a cabo en Github:

SGT 1 Login de usuario

Jaime Sarrion Sahuquillo edited this page 24 days ago · 4 revisions

Descripción

Como médico quiero registrarme y logearme para poder entrar a páginas restringidas sólo para mi. Incluir en la página de registro sólo los datos básicos: usuario, email y contraseña. El resto de datos del médico se introducirá en una página de actualización del perfil.

COS - Condiciones de satisfacción

Registro

- En el registro deberá pedirse: usuario, email, contraseña y ID

Nombre del campo	Descripción	Obligatorio	Criterios de aceptación	Especificación
Usuario	Nombre de usuario escogido por el usuario	Si	Debe ser único	Texto: máximo 20 caracteres
Email	Correo electrónico del usuario	Si		Texto: máximo 20 caracteres
Contraseña	Contraseña del usuario	Si		Texto: máximo 10 caracteres
ID	DNI del usuario	Si	Debe ser único	Texto: máximo 13 caracteres


Login

- Si el login ya existe se deberá dar un mensaje de error.
- En el login se pedirá el usuario y la contraseña. Si se introduce correctamente, se aparecerá un mensaje de bienvenida.

En este apartado, lo que hemos realizado es una descripción de la historia de usuario (como...quiero...para...). Además, le hemos añadido unas condiciones de satisfacción, sin las cuales no se dará la historia por resuelta. Finalmente, hemos añadido los Issues que componen la historia de usuario y que poco a poco la van completando. Como, por ejemplo:

Login de usuarios registrados #3

 Open JaimeSarrion opened this issue 24 days ago · 0 comments



JaimeSarrion commented 24 days ago

+


😊

⋮


Crear acciones, vistas y métodos de servicio para el login de usuarios registrados

- Formulario de login
- Acciones para mostrar el formulario y validarlo
- Capa de servicios que comprueba el login
- Test de login

Historia de usuario: [SGT-1 Login de usuario](#)




JaimeSarrion added this to To do in ARGOS PROYECT 24 days ago




JaimeSarrion self-assigned this 24 days ago

En este apartado, lo que se ha realizado es la creación de una rama nueva, para no interferir en la rama master, intentado aproximarse al comportamiento que se tendría con un equipo de desarrollo ágil. Una vez se ha completado la Issue, la persona que tiene asignada la Issue ha de hacer un PR o Pull Request:

Crear proyecto, modelo de médico y modelo de paciente #1

 Closed JaimeSarrion opened this issue 24 days ago · 0 comments · Fixed by #5



JaimeSarrion commented 24 days ago · edited

+


😊

⋮


- Crear el proyecto en React Native
- Crear un modelo de médico
- Crear modelo de cliente

Para la base de datos haremos uso de una en memoria, por ahora.


Historia de usuario: [SGT 1 Login de usuario](#)



JaimeSarrion added this to To do in ARGOS PROYECT 24 days ago




JaimeSarrion self-assigned this 24 days ago




JaimeSarrion referenced this issue 20 days ago

Creando proyecto #5


Merged



JaimeSarrion closed this in #5 20 days ago



JaimeSarrion moved this from To do to In PR in ARGOS PROYECT 19 days ago



JaimeSarrion moved this from In PR to Done in ARGOS PROYECT 19 days ago

Assignees

JaimeSarrion

Labels

None yet

Projects

Done in ARGOS PROYECT

Milestone

No milestone

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant

JaimeSarrion

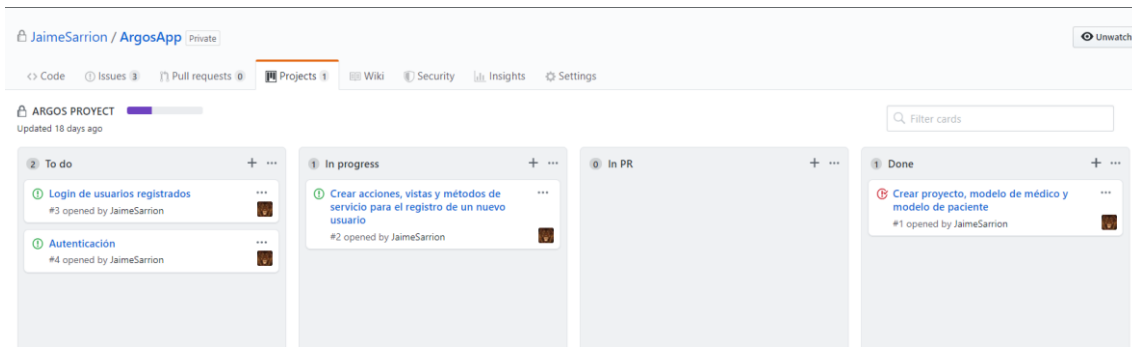
Lock conversation

Pin issue

Aquí lo que se está pidiendo es hacer un merge de la rama en la que estábamos con la rama master, por ello, alguien del equipo ha de echarle un ojo para comprobar

que no hay conflictos entre versiones y si está todo correcto, dar por bueno el PR y por tanto ‘mergear’ los cambios realizados.

Una vez se han realizado las historias de usuario pertinentes, Github nos brinda la posibilidad de crear un tablero en el que colocar las Issues y así ver en todo momento, en qué estado se encuentran.



Como se puede observar se ha optado por un modelo Kanban para la ordenación de las tareas, este está compuesto por 4 columnas: ‘To do’, ‘In progress’, ‘In PR’, ‘Done’. En la columna ‘To do’ tenemos las Issues que todavía no se han realizado y que por tanto están a la espera de que alguien se las asigne y las realice. En la columna ‘In progress’ están las Issues que han sido seleccionadas a alguien y este se ha puesto manos a la obra. En la columna ‘In PR’ tenemos las Issues, las cuales se quieren mergear con la rama master, pero antes tienen que ser revisadas por alguien para que no haya conflictos. En la última columna es donde residen las Issues que han sido completadas con éxito.

HERRAMIENTAS Y TECNOLOGÍAS

En este apartado me dispongo a mostrar y detallar las diferentes tecnologías que he ido utilizando a lo largo del proceso de desarrollo del proyecto. Además, voy a intentar aclarar por qué he escogido cada una de las tecnologías.

Las tecnologías escogidas han sido:

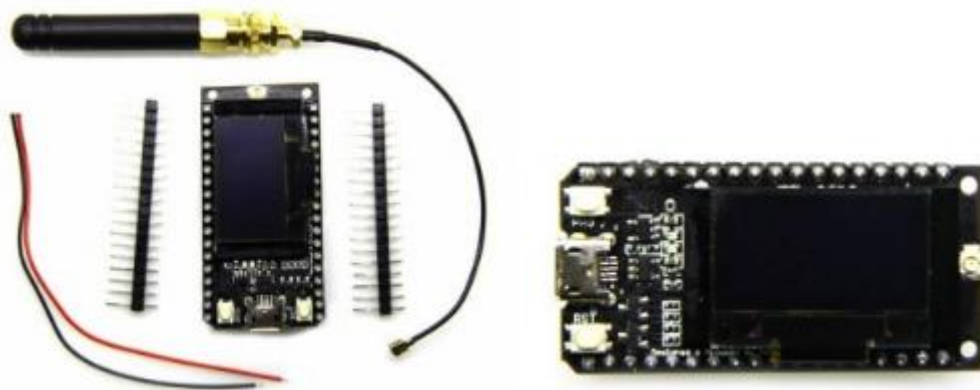
- **Hardware:** Raspberry, TTGO LoRa32 esp32 OLED, Arduino IDE
- **Diseño:** Photoshop
- **Software:** VSCode, Github, Trello, React, React Native, NodeJS, MySQL, CSS

Raspberry Pi

Cuando hablamos de Raspberry Pi, hablamos de una placa hardware que tiene los componentes básicos de un ordenador. Por lo que podemos decir que se trata de un ordenador en miniatura. Consta de una placa base sobre la que se monta un procesador, un chip gráfico y una memoria RAM, obviamente de limitadas prestaciones, pero no por ello deja de ser una herramienta realmente útil.

La razón por la cual he escogido Raspberry Pi es porque me parece una opción realmente económica de alojar mi API, además con mis limitados conocimientos en redes, he conseguido desplegar esa API directamente desde mi casa y sin ningún coste adicional, simplemente el coste inicial de comprar la Raspberry Pi. Por otro lado, si quisiera añadir más funcionalidades, además de alojar la API, podría ya que tiene recursos de sobra, sería posible incluso hacer que la API consumiera la base de datos alojada en la Raspy.

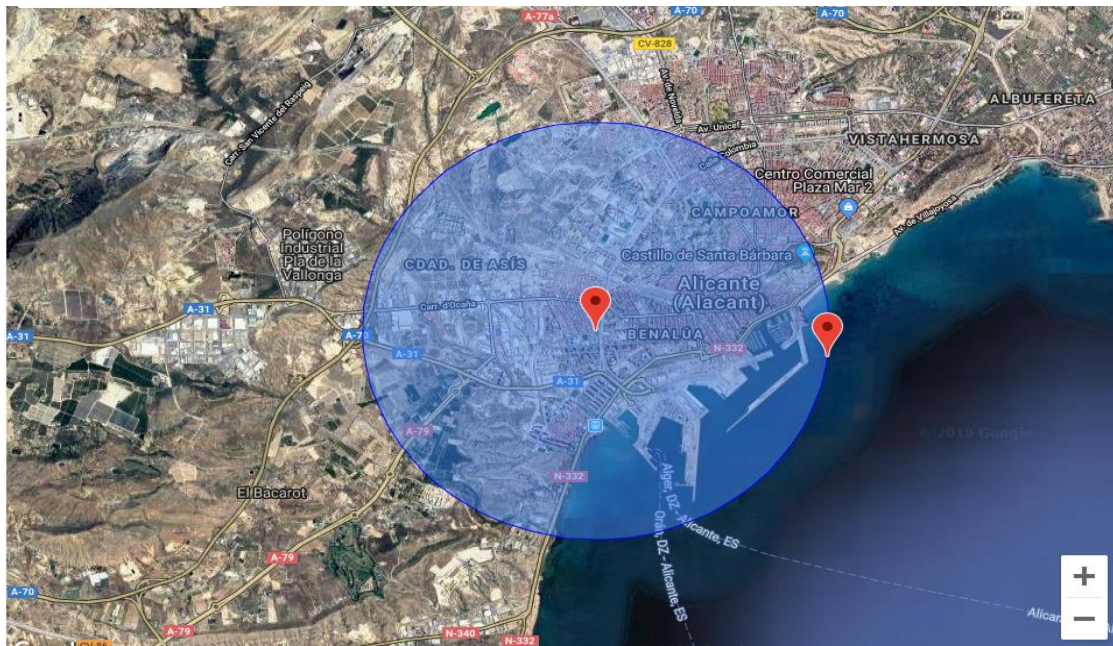
TTGO LoRa32 esp32 OLED



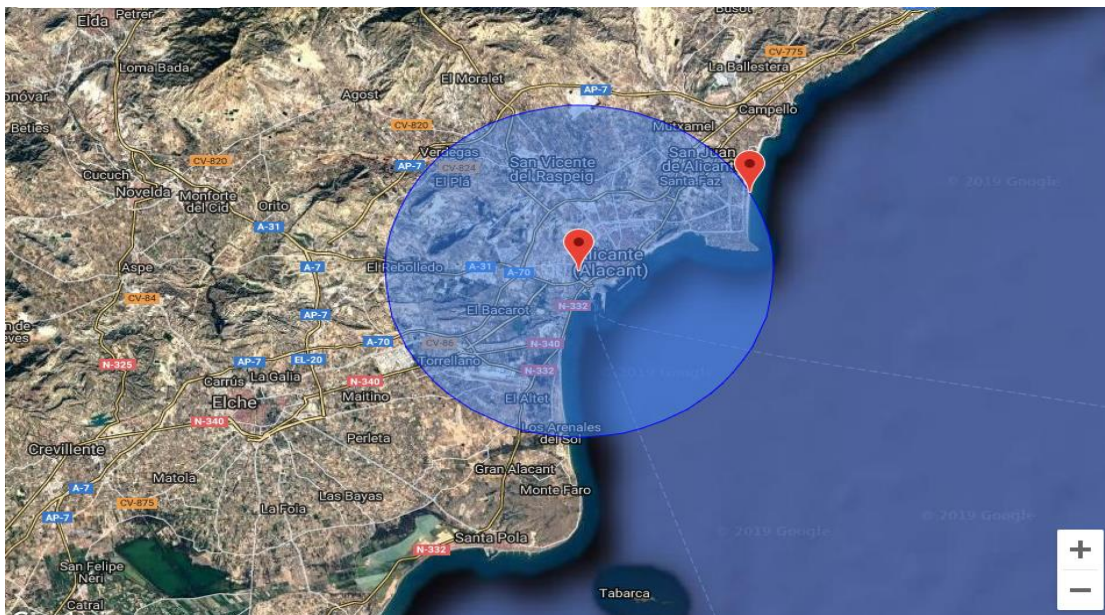
Esta es la placa elegida para su uso en este proyecto. Ésta, cuenta con un módulo Wifi compatible con Arduino y que ofrece conectividad inalámbrica 802.11b/g/n. Para un mejor manejo de la placa. Además, se le añadió una pantalla OLED que permite resaltar información que el usuario considere relevante, como en nuestro caso las pulsaciones que está registrando el sensor. Por otro lado, cuenta con un conector microUSB para la alimentación y que además permite programar la placa desde el IDE

de Arduino. Como guinda, presenta un módulo LoRa que permite transmitir de forma bidireccional datos a largas distancias. Este módulo en concreto tiene varias frecuencias de emisión, ya que dependiendo de la región donde te encuentres varía, la nuestra es la de 868Mhz, configurable por software.

Aquí se adjunta el alcance de la placa en una ciudad con un ruido eléctrico alto, por ello el radio de acción es de alrededor 3km:



Aquí se adjunta el alcance de la placa si se tratara de un entorno rural, al tener menos ruido eléctrico estaríamos hablando de 10km:



Adobe Photoshop



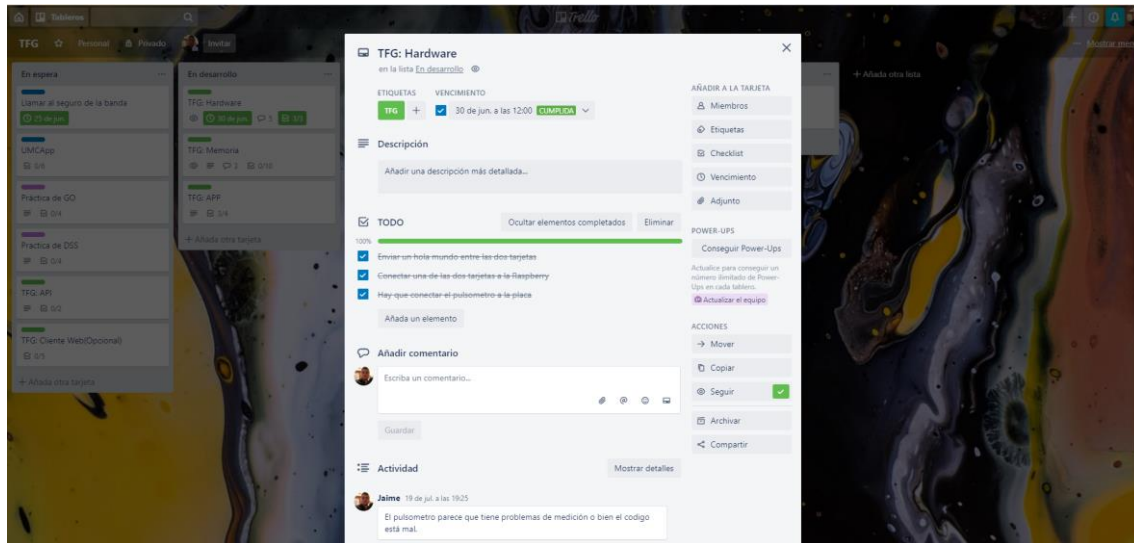
Desarrollado por [Adobe Systems](#), este programa esta accesible tanto para los usuarios de Windows, como para los de MacOS. Photoshop es una herramienta de edición de imagen de las más completas tanto para una persona experta, en el ámbito de la edición de imágenes, como para una persona que acaba de empezar. Esta, es una de las razones por las que he decidido usar Photoshop, ya que existen infinidad de tutoriales y blogs que te ayudan a aprender a usar esta herramienta. El uso que le he dado se ha limitado al diseño de la versión animada de Argos, así como el logo para la aplicación.

Trello



[Trello](#) es una herramienta de organización la cual es bastante simple, intuitiva y sencilla de usar. Esta herramienta emula a una pizarra, la cual podemos dividir en varias columnas, en mi caso la he dividido en: tareas pendientes, tareas en desarrollo, tareas de desarrollo acabadas, tareas por testear y tareas finalizadas. De esta manera, podemos ver con un simple vistazo, que es lo que tenemos por hacer, que es lo que estamos haciendo y que es lo que hemos terminado. Además de todo esto, podemos programar fechas de vencimiento, asignar tareas a las personas, las cuales están trabajando en el proyecto, crear checklist para ver el desarrollo de la tarea, etc. Todo esto de una manera muy sencilla e intuitiva, es por eso que Trello es capaz de competir con otros programas similares de pago, como lo es Jira. Trello no necesita ninguna formación para ser usado,

al contrario que Jira, que si requiere algunas nociones básicas para su uso.



La razón por la cual he escogido Trello como herramienta para organizarme, es porque ya tenía alguna experiencia usándola, es simple y muy visual. Además, me ha permitido realizar un modelo Kanban, usando un flujo de trabajo o workflow bastante natural, donde veía en todo momento, lo que me quedaba por hacer, lo que tenía hecho y lo que había terminado, esto, me permitía realizar estimaciones y ponerme fechas para las entregas.

Git

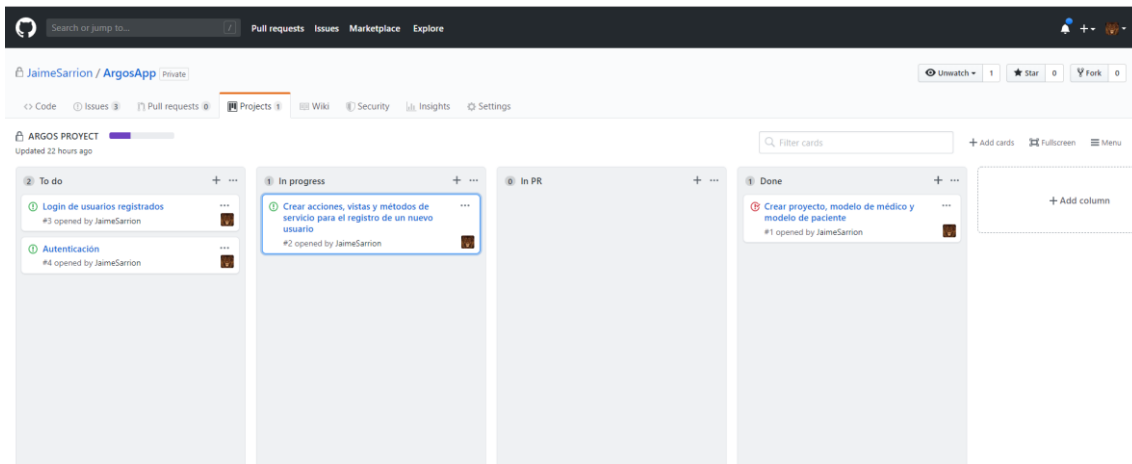


git

[Git](#) es la herramienta de control de versiones utilizada en el desarrollo tanto del API como de la aplicación. Anteriormente se hacía uso de los tradicionales CVS (Control de versiones). Este antiguo sistema de control de versiones, lo que realmente hacían era modelar la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo. A diferencia de estos sistemas de gestión de versiones, Git no modela ni almacena sus datos de este modo. Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que guardamos algo en Git se hace una instantánea de cómo está el archivo, para que, si vuelves a hacer otra “instantánea” y no has modificado algunos archivos, estos no se vuelven a subir, de esta manera se consigue optimizar todo el proceso de control de versiones.

La razón por la que he escogido Git como gestor de versiones de mi código, es porque hoy en día en casi todas las empresas se utiliza este sistema, y es una buena manera de tener una toma de contacto más a fondo que lo hecho durante la carrera.

Por otro lado, como servidor para alojar el código se ha escogido GitHub, el cual, te permite generar un Readme, una wiki, incluso un tablero para metodologías ágiles que está en todo momento actualizado, ya que, cuando alguien cierra una tarea, esta se mueve de una columna a otra.



Como conclusión extraída después de haberlo usado durante algún tiempo es que estamos ante una potentísima herramienta que no puede ser más útil. Además, las opciones que le han añadido para aportar información a aquella persona que quiera echarle un ojo a tu proyecto, hacen que sea bastante simple que esa persona pueda reutilizar tu código para algún proyecto suyo. Por otro lado, cuando se trabaja en grupo con esta herramienta, es increíble cómo se adapta al grupo y a sus tiempos, ya que, gracias a la creación de ramas, PR, Wiki y tablero, hace que cada programador pueda llevar un ritmo de programación sin necesidad de que nadie tenga que estar esperando a que alguien libere ficheros, que se están utilizando, como ocurre en los CVS.

Arduino IDE



[Arduino IDE](#) no es más que el entorno de desarrollo necesario para trabajar con cada una de las placas de Arduino. Quizás no es uno de los mejores editores de texto del

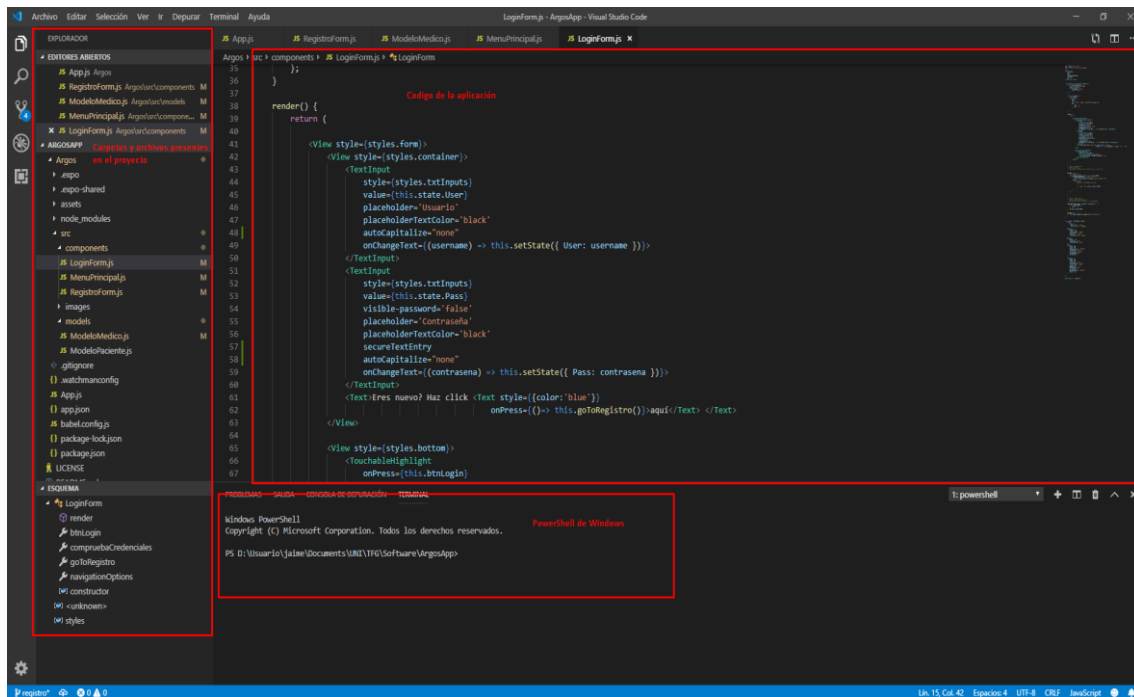
mercado, pero a la hora de programar placas es realmente útil debido a la gran comunidad que tiene detrás. Gracias a esta comunidad, podemos descargar infinidad de ejemplos para nuestras placas y con un simple clic ponerlo en marcha. Además de todo esto permite ver las E/S que le llegan a la placa, como por ejemplo la de los sensores de ritmo cardiaco, lo que te permite observar si el sensor está funcionando correctamente.

VSCode



[VSCode](#) o Visual Studio Code es un editor de código bastante completo, desarrollado por Microsoft para Windows, Linux y macOS. VSCode tiene presente una gran variedad de ventajas con respecto a otros editores, como, por ejemplo: Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, autoindentación. A todo lo anterior, se le añade que lo podemos personalizar de la manera que más nos guste, no sólo hablamos de cambiarle el color a la sintaxis, si no que podemos incluir plugins que nos generen código automáticamente, con tan solo escribir un par de palabras y darle a la tecla 'Enter'. Esta gran variedad de plugins puede deberse a que es un software gratuito y de código abierto, por lo que la comunidad puede crear los plugins que a ellos crean más útiles.

Este ha sido el editor que he escogido para el desarrollo tanto de la aplicación móvil, como de la API. Es un editor que es muy simple de usar, visual y funcional, además, de que presenta soporte para la gran mayoría de herramientas usadas para el proyecto, entre ellos react y react-native, es por ello, que he podido hacer uso de algunos de sus plugins, los cuales con escribir un par de palabras te generaban gran cantidad de código, esto se agradece cuanto tienes que escribir el mismo código varias veces, como por ejemplo a la hora de crear componentes o hacer un import.



React

[React](#) se define como una librería JavaScript centrada en el desarrollo de interfaces de usuario. Aun siendo así, React va más allá y se convierte en un perfecto aliado para el desarrollo de todo tipo de aplicaciones web, SPA (Single Page Application) o incluso para móviles. Para poder realizar todo esto, contamos con un completo ecosistema de módulos, herramientas y componentes, todo ello dirigido por el gestor de paquetes de node, el NPM (Node Package Manager) o bien también contamos con el gestor de paquetes desarrollado por Google y Facebook, este es yarn.

React-Native



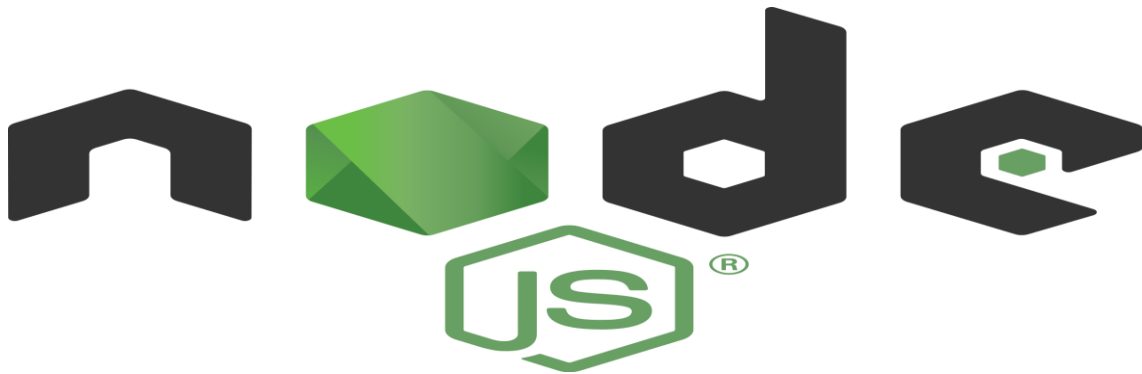
[React Native](#) es uno de los últimos frameworks que han salido al mercado de la mano de Facebook. Se trata de un framework de código abierto, el cual, te permite generar aplicaciones para Android, iOS, Web y UWP (Universal Windows Platform).

Hasta hace relativamente poco, había dos enfoques a la hora de desarrollar una aplicación, tenías que decidir si querías desarrollarlo de forma nativa o bien utilizar una plataforma web incrustada dentro de una webview o web app. Las ventajas de la forma nativa son claras, las aplicaciones están mejor optimizadas, el desarrollo se hace con unos

IDEs muy potentes (Android Studio, XCode...), además, presentan mejores compatibilidades con los sensores de los Smartphone. Por el otro lado, si elegíamos desarrollar a través de webview o web app, como por ejemplo Ionic, teníamos también muchas ventajas, como, por ejemplo, que no es necesario un conocimiento de desarrollo nativo de aplicaciones, sino, que con saber desarrollo web basta. Además, desarrollando una aplicación podemos tener soporte tanto en Android, iOS o UWP, eso es un ahorro de tiempo y dinero bastante considerable. Por contrapartida, podemos decir que las compatibilidades con Android, iOS y UWP muchas veces dan problemas y tienes que acabar programando específicamente para el dispositivo que despliegue la app.

¿Entonces, que pasa si quisiéramos las ventajas del desarrollo nativo, con las comodidades del desarrollo web? La respuesta es React Native. La ventaja de React Native es que la base es puro JavaScript, por lo que puedes aplicar tus conocimientos de React y JavaScript para crear una app de iOS y Android reutilizando el mismo código, manteniendo los componentes nativos para cada plataforma.

NodeJS



[NodeJS](#) es un entorno JavaScript desarrollado por Google, el cual tiene en mente la idea de que JavaScript no esté sólo en el lado del cliente, sino que también, pueda ser utilizado en el lado del servidor. Gracias a NodeJS los servidores son capaces tener un entorno de ejecución en el que compilan y ejecutan JavaScript a unas velocidades altísimas. Por lo tanto, podemos decir que Node es una librería y un entorno de ejecución de E/S asíncrona y dirigida por eventos.

La escalabilidad es uno de los objetivos principales que Google ha conseguido con Node y es que gracias a este los servidores pueden tener un número elevado de conexiones de forma simultánea con el servidor. Esta es la principal razón por la cual he escogido Node para la creación de la API. Me pareció una buena oportunidad para adentrarme más en el mundo de Node, más allá de lo que hayamos podido ver en clase.

MySQL



[MySQL](#) es uno de los sistemas de gestión de bases de datos relacionales más famosos de todo el mundo. Por una parte, es de código abierto, pero también, cuenta con una versión gestionada por Oracle. Una de las principales características de MySQL es que trabaja con bases de datos relacionales, esto quiere decir, que utilizan tablas de datos múltiples que se conectan entre sí a través de relaciones para organizar y almacenar la información correctamente.

Para el proyecto he decidido usar una base de datos relacional en lugar de una no relacional, como puede ser [MongoDB](#), por el simple hecho de que, a la hora de escalar la aplicación, las bases de datos no relacionales, presentan más problemas. Además, MySQL tiene opción para crear unas funciones que llevan a cabo operaciones y estas dependiendo de la frecuencia con la que se realizan estas operaciones, se van optimizando poco a poco.

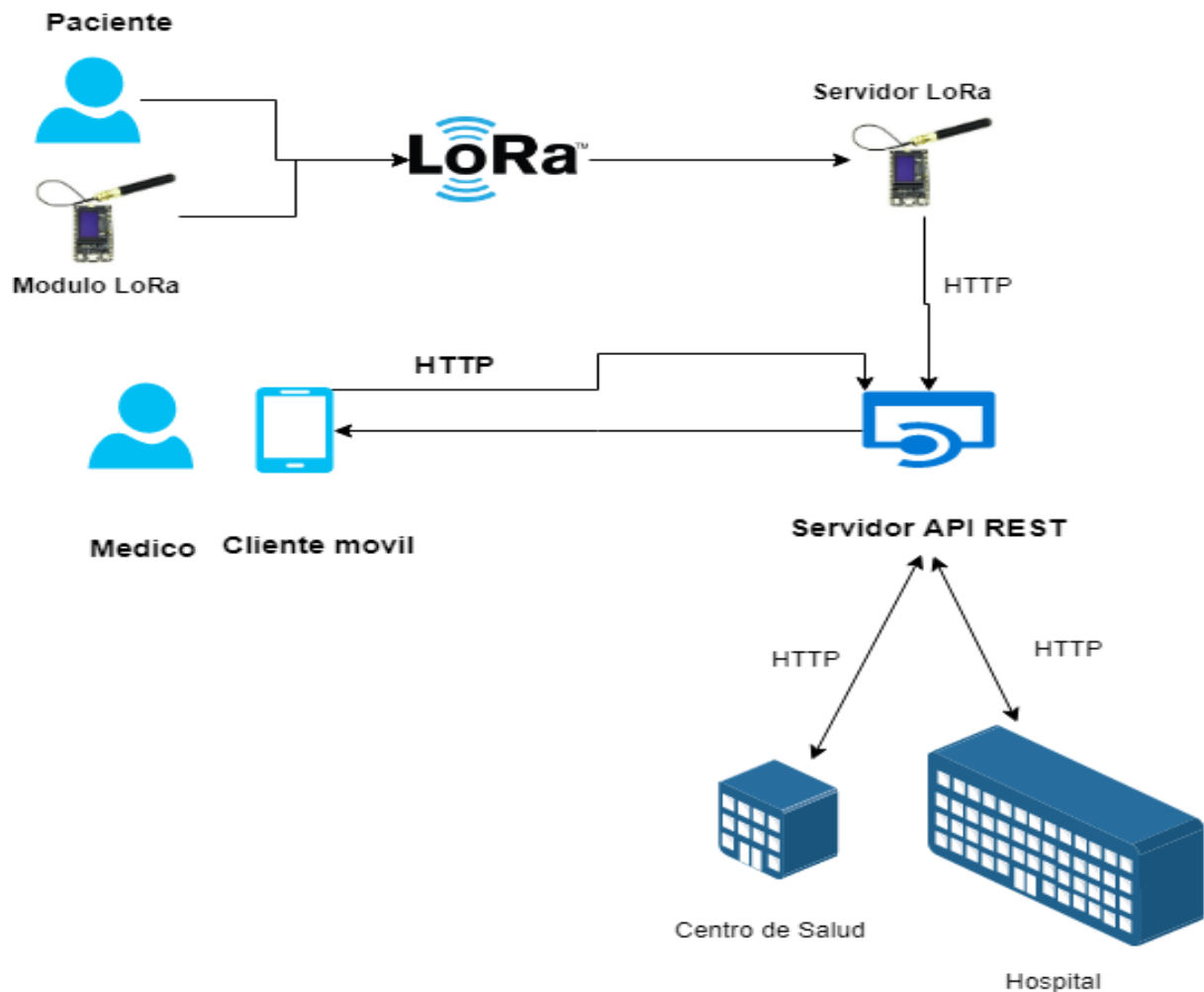
CSS



[CSS](#) o Cascading Style Sheets, es un lenguaje de estilo gráfico, la función del cual es establecer un diseño visual en los documentos web o en nuestro caso en la aplicación móvil, ya que React Native hace uso de CSS para el diseño gráfico de la aplicación. CSS está diseñado principalmente para marcar las separaciones del contenido del documento, aunque también es usado para darle color, animaciones y otras muchas acciones que este lenguaje nos permite. Una de las principales ventajas de este lenguaje,

es que nos permite compartir hojas de estilo para diferentes páginas, por lo que nos ahorra la repetición de código.

ARQUITECTURA DEL SISTEMA



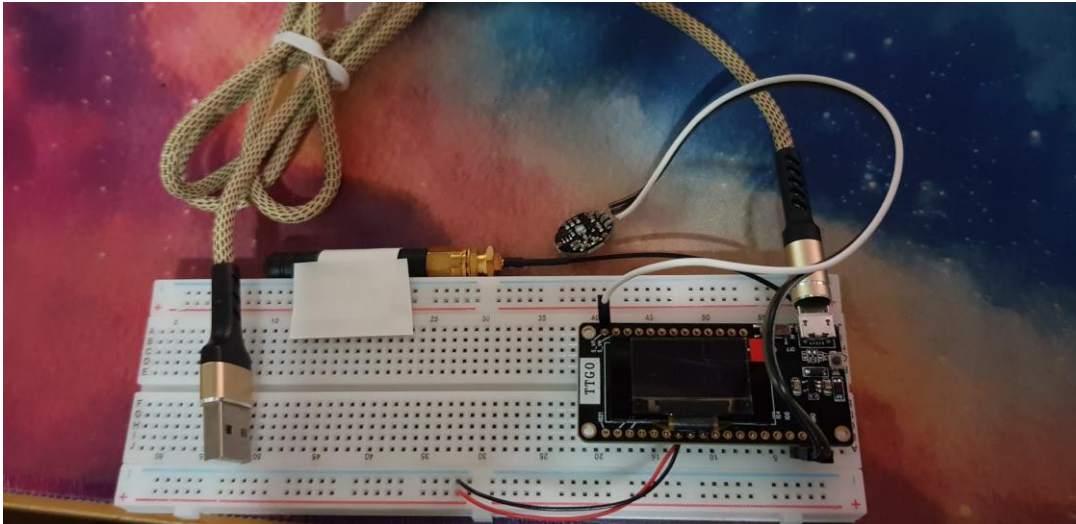
En esta imagen se representa la arquitectura diseñada para este proyecto de monitorización de personas en riesgo.

Lo que se quiere hacer con este proyecto es conectar una serie de sensores al paciente, de la manera menos invasiva posible. Estos sensores se comunicarían con una placa que es capaz de transmitir vía LoRa. A continuación, una placa instalada como servidor, en un centro de salud, recoge los datos de los sensores y los manda a la API a través de peticiones HTTP. Esta API en caso de que reciba una señal anómala de los sensores, dará parte al hospital o centros de salud pertinentes para que estén alerta. Por otro lado, los profesionales de la salud, tendrán acceso a un cliente móvil, con el que podrán ver un registro de las constantes vitales del paciente.

Modulo LoRa Cliente

Repositorio del cliente: <https://github.com/JaimeSarrion/ArgosHardware>

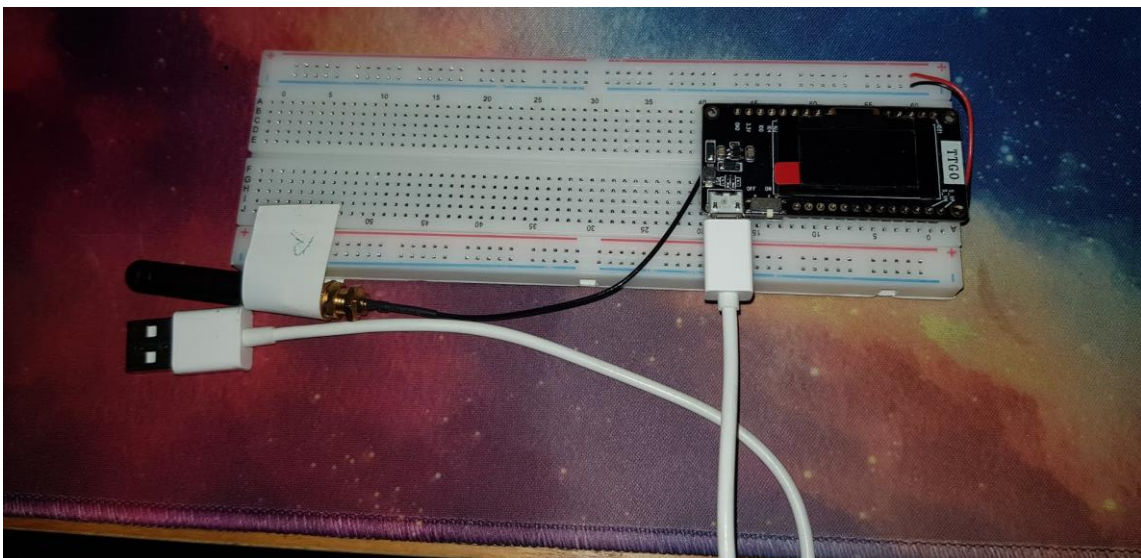
Este módulo es el que acompañará al paciente en todo momento y al que estarán conectados todos los sensores necesarios para la monitorización de las constantes vitales del paciente.



Como podemos observar, tenemos una protoboard con la placa y el sensor montados. En este caso estamos utilizando la placa TTGO LoRa32 ESP32 con OLED la cual tiene Wifi, bluetooth y LoRa integrados, pero lo ideal es que solo tuviera LoRa para poder reducir su tamaño considerablemente y que así la antena y los sensores pudieran estar en algo tan liviano como una pulsera.

Modulo LoRa Servidor

Repositorio del servidor: <https://github.com/JaimeSarrion/ArgosHardware>



Aquí podemos observar el modulo que se ha programado como servidor, el cual no necesita más que una pequeña antena, y un conector microUSB del cual podríamos prescindir, ya que, la placa viene con dos cables (negro y rojo, en la foto), los cuales podemos conectar a una batería para alimentar a la placa. En este caso, esta placa es perfecta ya que podemos conectarnos a la red Wifi de donde esté instalada y así enviar los datos recibidos a la API correspondiente.

Servidor API REST

Repositorio de la API: <https://github.com/JaimeSarrion/ArgosAPI>



La API se encuentra desplegada en la Raspberry con el sistema operativo Raspbian, el cual es bastante similar a Debian, lo que hace que sea realmente sencillo de manejar. Lo único que tenemos que hacer es conseguir tener un dominio que apunte a nuestra IP y luego redireccionar el puerto que queramos del router a la IP fija de la Raspberry. Este proceso se explicará más adelante.

Esta API es la encargada de gestionar las peticiones que le llegan, entre sus funciones están:

- Guardar los datos de los sensores y usuarios en BD
- Generar alertas en caso de datos anómalos
- Devolver datos al cliente móvil

Cliente móvil

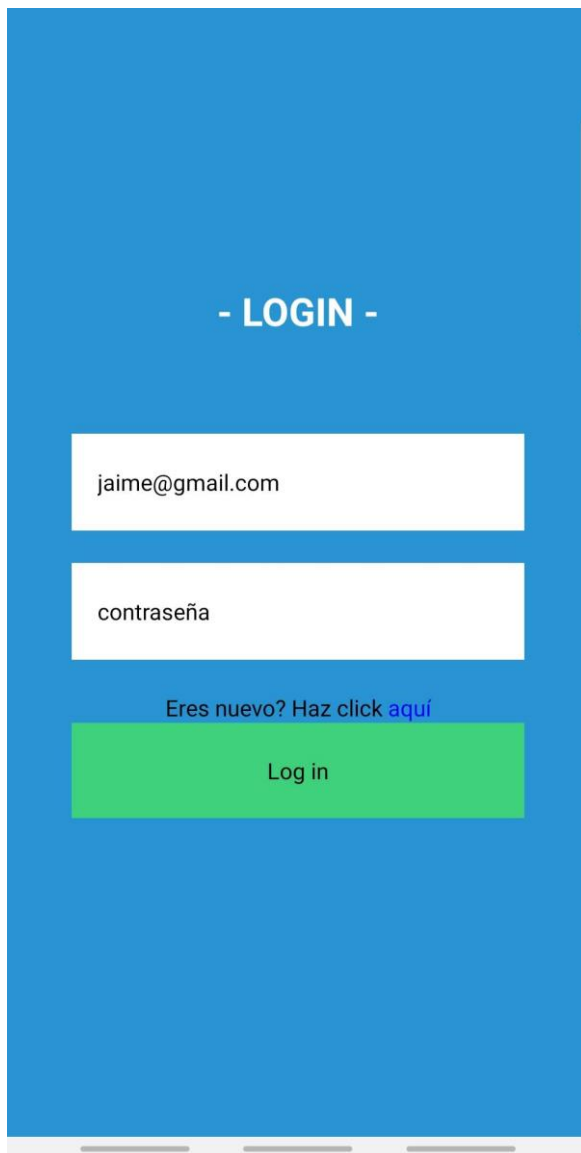
Repositorio del cliente móvil: <https://github.com/JaimeSarrion/ArgosApp>

En este proyecto se ha implementado un cliente para móvil a modo de ejemplo de lo que se podría llegar a conseguir. Este cliente ha sido realizado utilizando React Native, el cual es una excelente opción para programar realizar una aplicación y esta sirva tanto para Android como para iOS.

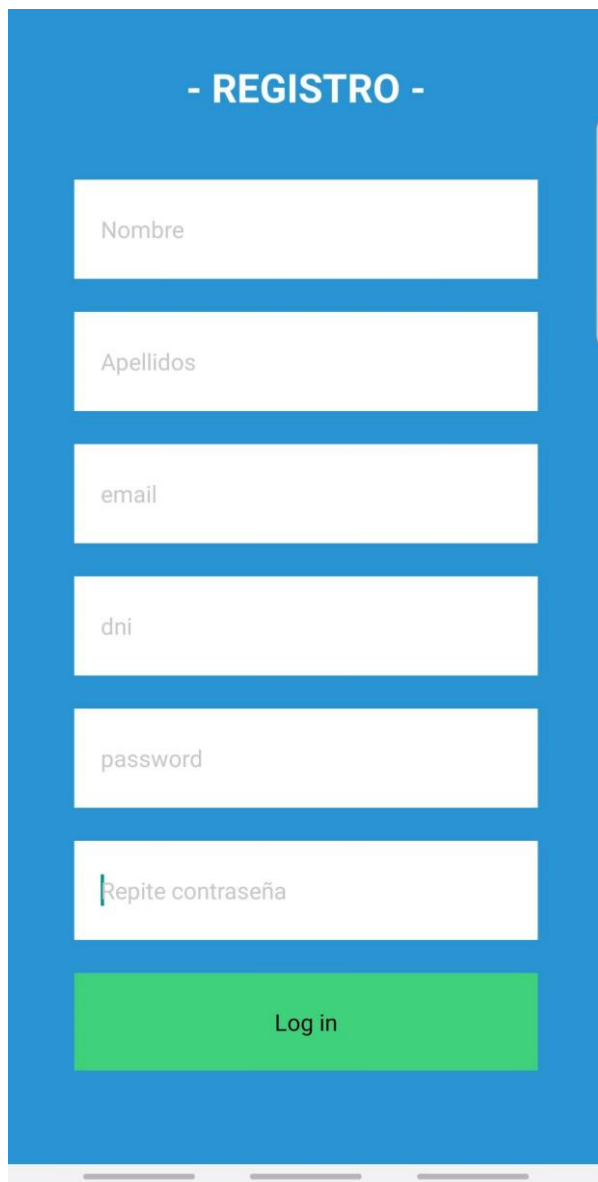
El cliente móvil nos permite:

- Hacer login como médico
- Registrarnos como médico
- Ver un listado de pacientes
- Acceder a los detalles del paciente
- Ver sus constantes vitales en directo

Estas son algunas de las pantallas que realizan las acciones anteriormente citadas:

A mobile app login screen with a solid blue background. At the top center, the text "- LOGIN -" is displayed in white. Below this, there are two white rectangular input fields. The first field contains the text "jaime@gmail.com". The second field contains the text "contraseña". Below the second field, there is a line of text: "Eres nuevo? Haz click [aquí](#)", where "aquí" is a blue hyperlink. At the bottom of the form area, there is a green rectangular button with the text "Log in" in black. The entire screen is framed by a thin white border at the bottom, suggesting a mobile device interface.

Como podemos observar, aquí tenemos la típica pantalla de login donde el médico simplemente ha de introducir el email y la contraseña y presionar el botón de login. En caso de que los datos sean incorrectos, o que el usuario no se encuentre en la BD saltará un mensaje indicándolo.



- REGISTRO -

Nombre

Apellidos

email

dni

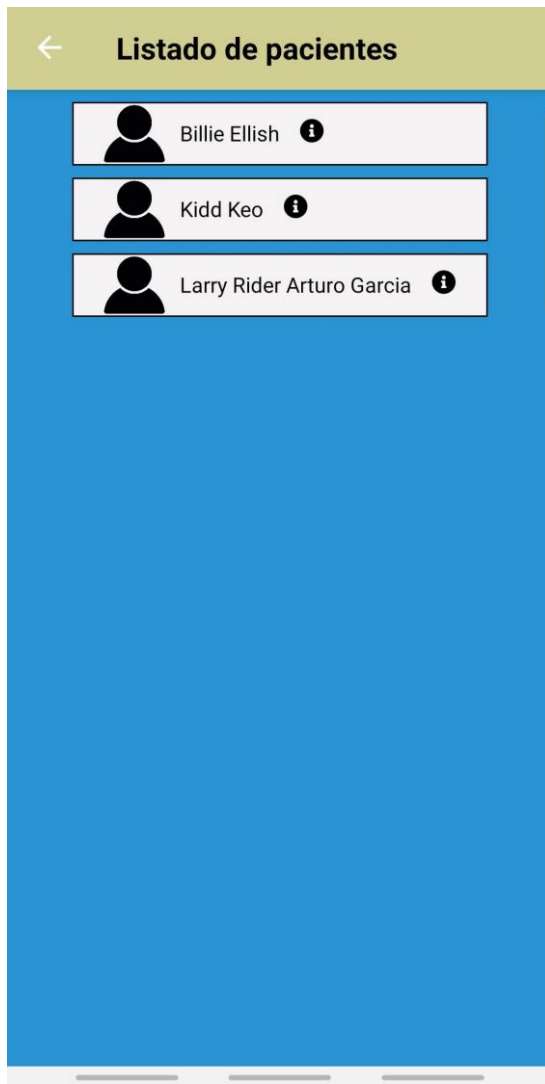
password

Repita contraseña

Log in

The image shows a mobile application registration screen. It has a solid blue background. At the top, the title '- REGISTRO -' is centered in white. Below the title are six white rectangular input fields, each with a light gray placeholder text: 'Nombre', 'Apellidos', 'email', 'dni', 'password', and 'Repita contraseña'. The last field has a small green vertical bar at the start of the text. At the bottom of the form is a green rectangular button with the text 'Log in' in black. The entire form is centered on the screen. At the very bottom, there are three thin horizontal lines representing the mobile home indicator.

En caso de no estar registrado, deberá de hacerlo a través de esta pantalla, donde tendrá que introducir algunos datos necesarios.



Aquí tenemos la ventana donde nos aparece el listado de pacientes que tiene el médico, en caso de pinchar en uno de ellos, nos mostraría los detalles de este, como constantes vitales, observaciones, etc.

PROCESO DE DESARROLLO

En este apartado me dispongo a exponer como ha sido el desarrollo del proyecto. Como se podrá observar he cambiado la narración a primera persona, me ha parecido adecuado darle un enfoque más personal, ya que describo como ha sido mi desarrollo en el proyecto. Este proceso lo he dividido en:

Búsqueda de información:

- ¿Qué es LoRa y sus características?
- ¿Qué es LoRaWan
- Clases de dispositivos
- Elementos necesarios para el desarrollo del proyecto

Hardware:

- Hola Mundo en LoRa
- Transmitir un mensaje entre las placas
- Como conectar una de las placas a la Raspberry Pi
- Montaje de la API en la Raspberry
- Conexión del sensor de ritmo cardiaco

Software:

- Creación de mockups
- Diseño de Argos
- Diseño de la BD
- Diseño del API

Búsqueda de información

Todo proyecto que se precie necesita tener una previa búsqueda de información de que es lo que se va a hacer, cómo y por qué, esto ayuda a reducir los errores y poder seguir un camino a la hora del desarrollo del proyecto. En mi caso, realicé una búsqueda sobre una nueva tecnología Wireless y sus posibles aplicaciones, LoRa. Por otro lado, también estuve buscando otros proyectos IoT que la comunidad había hecho para ver realmente hasta donde había llegado la gente con esta nueva tecnología.

¿Qué es LoRa y sus características?

LoRa se desarrolló en Francia por Cycleo, más tarde, fue adquirida por la empresa llamada [Semtech](#) en 2012 y que actualmente tiene la patente. Semtech, fabrica los chips de radio o cede la propiedad intelectual a otras empresas fabricantes de chips, como, por ejemplo, [Microchip](#). LoRa pertenece a la tecnología LPWAN (Low Power Wide Area Network) y tiene las siguientes características:

- **Conectividad de larga distancia:** Conecta dispositivos de hasta 48 km de distancia en áreas rurales y penetra en densos ambientes urbanos o interiores profundos.

- **Conectividad segura mediante encriptación:** soporta la encriptación AES128, de extremo a extremo, autenticación mutua, protección de integridad y confidencialidad.
- **Envío bidireccional de paquetes de datos.**
- **Consumo eléctrico ultra mínimo, evitando así el consumo de energía:** Requiere energía mínima, con una vida útil prolongada de la batería de hasta 10 años, lo que minimiza los costos de reemplazo de la batería.
- **Opera en la banda ISM (Industrial Scientific and Medical) de radio:** la cual está habilitada en todo el mundo para uso no comercial y no necesita licencia.
- **Las frecuencias varían según la región:** En Europa se utiliza 868 MHz y 433, por otro lado, en USA se utiliza la frecuencia de 915Hz.
- **Actualmente se está usando para:** proyectos de gestión de energía, control de contaminación, estructuras eficientes, prevención de desastres y mucho más.
- **Geolocalización:** Permite aplicaciones de rastreo sin GPS, ofreciendo beneficios únicos de baja potencia que otras tecnologías no han tocado.
- **Móvil:** Mantiene la comunicación con dispositivos en movimiento sin afectar el consumo de energía.
- **Chirp Spread Spectrum modulation(CSS):** modula los datos sobre diferentes canales de frecuencia y velocidad.
- **Velocidad:** de 0.25 Kbps a 11Kbps. A máximo alcance, menor velocidad. Además, se le puede ajustar el spreading factor, para enviar datos redundantes.
- **Baja tasa de datos:** está diseñado para pequeñas cantidades de datos de dispositivos.

¿Qué es LoRa WAN?

LoRa Wan es un protocolo de comunicación en las redes LPWAN. Se puede considerar como el equivalente de las capas 2 y 3 del modelo OSI (Enlace de Datos y Red, consecutivamente). Mientras que LoRa es la capa física de acceso a la red (la parte de radiofrecuencia) LoRa WAN define el protocolo de acceso a red y la arquitectura del sistema.

LoRa Alliance es una asociación creada en 2015 abierta y sin ánimo de lucro, actualmente tiene más de 500 miembros, algunos de ellos muy importantes en el mundo de las telecomunicaciones como: CISCO, IBM, Orange, Semtech, Microchip, ect. La misión de esta asociación es la de estandarizar el protocolo LoRa WAN y garantizar la interoperabilidad de todos los productos y tecnologías de LoRa WAN.

Clases de dispositivos

En Lora WAN los dispositivos pueden funcionar de tres maneras diferentes:

- **Clase A:** la más soportada en casi todos los dispositivos, este tipo de clase ofrece el mayor ahorro de energía debido a que sólo entra en modo escucha después de enviar un dato hacia un Gateway, por eso, es ideal para dispositivos que usan una batería.

- **Clase B:** las ventanas de recepción son establecidas en unos tiempos determinados.
- **Clase C:** estos dispositivos tienen su radio encendida casi todo el tiempo, la única vez que no pueden recibir es cuando están transmitiendo un mensaje. Está diseñado para dispositivos alimentados por la red, como un actuador que necesita poder controlar en cualquier momento.

Elementos necesarios para el desarrollo del proyecto

Una vez había decidido que quería realizar el proyecto con LoRa, una pregunta me surgió a la cabeza: ¿Cómo empiezo? Pues bien, la respuesta la hallé en un foro donde explicaban como hacer un Gateway con un módulo LoRa, ahí hallé un trozo de hilo del que tirar. Comencé a investigar sobre ese modulo (la placa TTGO LoRA32 ESP32 OLED) y resulta que la gente había hecho varios proyectos IoT con él, por lo que tenía que ser apto para el proyecto que quería realizar. Una vez localizado el modulo, las demás partes más o menos las tenía claras: necesitaba dos placas LoRa que se comunicaran entre ellas, un servidor donde alojar la API y una posible base de datos, un sensor de ritmo cardiaco el cual pueda conectar a una de las placas.

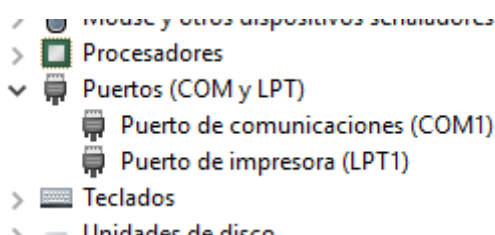
Una vez tenía claro, más o menos, lo que necesitaba de hardware, tenía que ver como programar ese hardware, ya que no lo había hecho en mi vida. Por suerte, encontré varios tutoriales que mostraban ejemplos del famoso hola mundo en Arduino IDE.

Hardware

Una vez seleccionados los componentes y hallada la forma de programarlos era el turno de empezar con las pruebas, a ver si conseguía hacer lo que aparecía en los tutoriales y foros. Estos son algunas de las fases por las que pasé en el desarrollo del código del hardware:

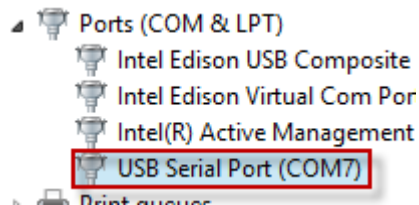
Hola mundo en LoRa

El objetivo era claro, intentar mostrar un mensaje en la consola de “Hola mundo”, enviado por la placa. Parece algo sencillo, pero cuando me puse a programarlo resultó no serlo tanto. Mi principal problema es que todos los ordenadores a los que conectaba la placa, no la reconocían. En un principio, solo tenía que instalar los drivers, instalar Arduino IDE y configurarlo para programar con las placas TTGO LoRa, pero me aparecía lo siguiente en los dispositivos:



Lo normal es que, al conectar la placa, aparezca en los puertos COM y LPT, algo así como COM7 para posteriormente configurar el IDE de Arduino para que se

comunique con la placa a través de ese puerto, pero no aparecía. Investigando, di con cantidad de drivers para estos puertos, incluso con programas que analizaban el estado de los puertos, los cuales parecían estar bien. Mi última opción, era pensar que el ordenador estaba bien, y efectivamente, si no detectamos la placa al conectarla, puede ser que el error este o bien en la placa o bien en el cable que la conecta al ordenador, como fue mi caso. Al cambiar el cable, por fin apareció en dispositivos:



A continuación, me dispuse a realizar el 'hola mundo', para ello descargue una librería llamada "[LoRa Sandeep Mistry](#)". Una vez hecho esto, podemos abrir un proyecto de los que vienen de ejemplo o bien crearlo nosotros, lo recomendable es empezar con uno sencillo que venga ya más o menos configurado. Lo primero que tenemos que hacer es incluir las librerías, en mi caso, estas:

```
#include <SPI.h>
#include <LoRa.h>
```

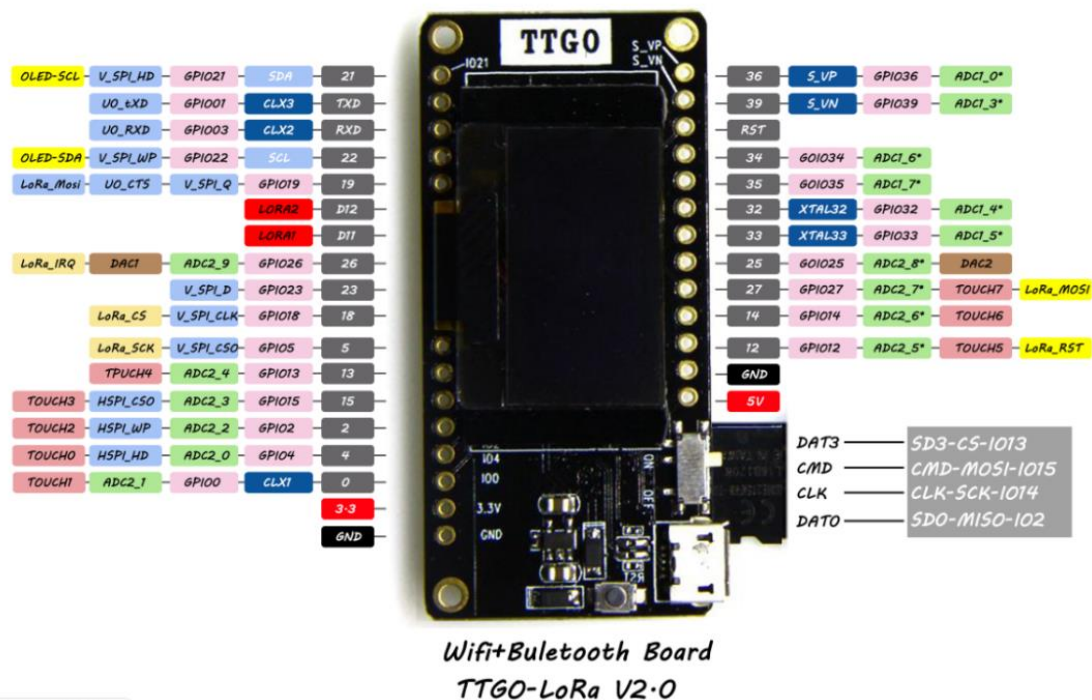
Una vez hecho esto, tenemos que configurar el método setup. Antes de nada, hay que saber que en Arduino hay dos métodos principales que son setup y loop. En el método setup, tenemos que poner las configuraciones que queremos que se hagan antes de empezar con el programa. En cambio, el método loop, como su nombre indica, es un método que no para de repetirse, aquí es donde tenemos que poner las acciones que queremos que se realicen en la placa, como, por ejemplo, que se conecte a la Wifi disponible. Una vez sabido esto, vamos a configurar el setup, para nuestro 'hola mundo'.

```
void setup() {
  Serial.begin(9600);
  while (!Serial);

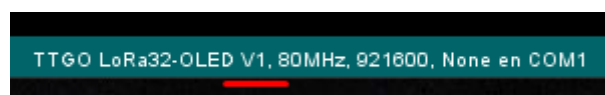
  Serial.println("LoRa Sender");
  LoRa.setPins(18,12,26);

  if (!LoRa.begin(868E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}
```

En la primera línea le estamos diciendo a que baudios queremos que inicie el Serial, esto es para poder abrir una ventana desde Arduino IDE, configurar esos baudios y poder ver las salidas que ejecuta la placa. Otra línea importante, que no puede faltar, es en la que configuramos los pines LORACS, RESET e IRQ, en nuestro caso 18,12 y 26 respectivamente. Lo podemos comprobar en la ficha técnica de la placa:



Esta configuración es necesaria para el funcionamiento del programa, además hay que tener cuidado con los números si hemos decidido abrir un ejemplo de los que vienen en la librería, ya que, por defecto vienen los pines de la versión 1.0 de la placa, en mi caso, al ser la versión 2.0 la ubicación de los pines es distinta y por ello los números también, esto lo podemos ver en la esquina inferior derecha de Arduino IDE.



Otra de las líneas importantes de la función de setup, es la de:

```
if (!LoRa.begin(868E6)) {
  Serial.println("Starting LoRa failed!");
  while (1);
}
```

En esta línea le estamos diciendo en que frecuencia tiene que emitir, es muy importante emitir en la frecuencia correcta, puesto que, si emitimos en la frecuencia equivocada estaremos infringiendo la ley. Además, hay que resaltar que el tiempo

tiempo que puedes estar emitiendo según la ley es muy reducido, en nuestro caso, para no infringir la ley, tenemos que emitir una vez cada 3 minutos. Lo que nos dice la ley es que no podemos transmitir más que el 1% del tiempo. Es decir, si enviar un paquete nos lleva 100ms, tendremos que mantenernos callados 900 ms.

```
void loop() {  
  
    Serial.println("Hola mundo");
```

Finalmente, en el módulo de loop (), si queremos mostrar por el Serial (configurado a 9600 baudios) el mensaje: “Hola mundo”. Simplemente hemos de hacer un println y añadirle el mensaje que queremos que muestre por el terminal.

Transmitir un mensaje entre placas

Una vez conseguí hacer funcionar una de las placas, el siguiente paso era obvio, tenía que enviar un ‘hola mundo’, de una placa a otra. Para sorpresa mía, resulta que la librería de LoRa lo hace realmente sencillo:

```
// send packet  
LoRa.beginPacket();  
LoRa.print("hello ");  
LoRa.print(counter);  
LoRa.endPacket();
```

Lo que hacemos con esta librería es, en un primer momento, crear el paquete, con “.beginPacket()”. A continuación, le añadimos información a ese paquete, como puede ser un par de cadenas de texto. Finalmente, con el “endPacket”, enviamos el paquete. En este caso, no estamos enviando a nadie en concreto, simplemente a quien pueda recibirlo. Ahora, solo nos hace falta programar la otra placa, para que pueda recibir mensajes:

```

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet ");

    // read packet
    while (LoRa.available()) {
      Serial.print((char)LoRa.read());
    }

    // print RSSI of packet
    Serial.print(" with RSSI ");
    Serial.println(LoRa.packetRssi());
  }
}

```

Para el caso del receptor, simplemente hemos de llamar a la función “parsePacket” que es la encargada devolver 0 o 1, dependiendo de si ha recibido algo o no. A continuación, cuando se ha recibido un paquete, lo que intenta es leerlo e imprimirlo por pantalla. Gracias a esta librería, enviar y recibir un mensaje se hace realmente fácil, aunque estaría bien poder enviarlo a una placa en concreto. De todas formas, lo que podemos hacer es cifrar el mensaje que queramos enviar, para que, si alguien lo recibe, no sepa que es lo que es.

Como conectar una de las placas con la Raspberry

Una vez pude transmitir información entre las dos placas, tocaba conectarse con la Raspberry. En un primer momento, mi objetivo era conectar unos cables de la Raspberry Pi a la placa que hace de receptor y que esta le enviara los datos a la ‘Raspy’. Más tarde, me di cuenta que lo más sencillo era conectar la placa receptora a una red Wifi y con ella, hacer peticiones HTTP a la API alojada en la ‘Raspy’. Para conectar la placa y enviar peticiones http, necesitamos incluir 2 librerías más:

```

#include <SPI.h>
#include <LoRa.h>
#include <WiFi.h>
#include <HttpClient.h>

```

Una vez añadidas tenemos que añadir unas líneas más en la sección de setup:


```

void setup() {

  Serial.begin(9600);
  while (!Serial);

  delay(4000);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  Serial.println("Connected to the WiFi network");

  Serial.println("LoRa Receiver");
  LoRa.setPins(18,14,26);

  if (!LoRa.begin(868E6)) { //Es la frecuencia de europa
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

```

Gracias a la librería “Wifi.h” el proceso para conectarse a una wifi doméstica se convierte en algo realmente sencillo, tanto que con tan solo 3 líneas somos capaces de conectarnos a una red wifi doméstica. Lo que estamos haciendo básicamente es introducir el nombre de la red y la contraseña a la que nos queremos conectar. Antes de eso, le hemos introducido un delay para que no nos bloquee el router por hacer muchos intentos de conexión seguidos. Una vez hemos introducido el SSID y la contraseña comprobamos si estamos conectados, cuando esto ocurra, se mostrará un mensaje en la consola que nos dirá que se ha conectado correctamente, y ya está, ya estamos dentro. Ahora, vamos con las peticiones HTTP:

```

    if ((WiFi.status() == WL_CONNECTED)) { //Check the current connection status
      HTTPClient http;

      http.begin("http://jsonplaceholder.typicode.com/posts"); //Specify the URL
      http.addHeader("Content-Type", "text/plain");
      int httpResponse = http.POST(packet); //Make the request

      if (httpResponse > 0) { //Check for the returning code
        String response = http.getString();
        Serial.println(httpResponse);
        Serial.println(response);
      }
      else {
        Serial.println("Error on HTTP request");
      }
      http.end(); //Free the resources
    }
}

```

Aquí tenemos un ejemplo de una petición común HTTP. Primeramente, lo que hacemos es verificar que estamos conectados a la red WiFi y que por tanto somos capaces de enviar y recibir datos. A continuación, creamos un cliente HTTP con el que podremos crear un paquete, con su cabecera y su cuerpo. En este caso, vamos a hacer una petición post y vamos a enviar las pulsaciones recibidas de la placa emisora. Para ello, tenemos que especificar a que url queremos enviar los datos. Luego, si es necesario, añadirle alguna cabecera y posteriormente enviarlo, añadiéndole los datos en el body con la función `$. Post()`. Gracias a la librería `“HTTPClient.h”` hacer peticiones se vuelve realmente sencillo, tanto que con tan sólo 4 líneas hemos sido capaces de comunicar los datos necesarios a la API.

Montaje de la API en la Raspberry

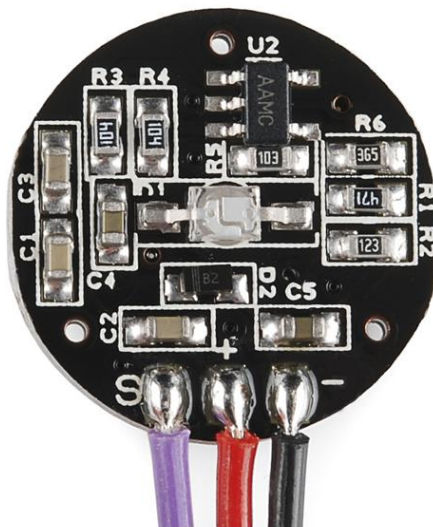
Para el montaje de la API en la Raspberry tuve varias ideas, una de ellas fue encontrar un servidor donde alojar la API gratuitamente, por ello elegí Heroku. Heroku nos brinda la posibilidad de alojar una pequeña API e incluso que pueda hacer uso de una base de datos, eso sí, todo ello con bastantes limitaciones, ya que es un servicio gratuito. Aunque no terminé usando Heroku, voy a explicar lo que hice para subir el proyecto, ya que me llevo algún tiempo.

Para subir un proyecto a Heroku, lo primero que tenemos que hacer es instalar el cliente de Heroku en el ordenador. A continuación, tenemos que ejecutar el comando `‘heroku login’` el cual nos pedirá las credenciales para el login. Una vez loggeados, tenemos que `‘crear’` una app de Heroku con el comando `‘heroku create’`. Una vez termina de crear la app, tenemos que enviar nuestro código al repositorio de Heroku, esto lo podemos hacer a través del comando `‘git heroku master’`. Una vez finalizado el push, ya tenemos nuestra aplicación desplegada y funcionando, sólo falta ver cuál es la URL a la que tenemos que hacer las peticiones, esto se ve desde el panel de control de Heroku, donde aparecen todos los proyectos que tenemos desplegados.

Una vez lo tenía ya todo listo en Heroku, haciendo pruebas me di cuenta de que podía tener el servidor alojado en mi casa, sin limitaciones de ningún tipo. Lo único que necesitaba conseguir era un dominio, a poder ser gratuito, como, por ejemplo, los que te da la página de [FreeMyIP](https://freemyip.com). Lo que hace esta página es generarte un dominio gratuito, con la terminación `“. freemyip.com”`, el cual se asocia a la IP que tú le digas. Para configurar la IP tienes que hacer una petición GET a un link que te da la propia página y entonces asocia el dominio con la dirección IP, como resultado tienes un dominio gratuito al que poder hacer peticiones si tienes una API desplegada, como es en este caso. Por último, lo que hay que hacer es elegir dónde quieres alojar la API (en este caso la Raspberry), hacer que ese alojamiento tenga una IP fija y posteriormente en el router, lo configuramos para que las peticiones que vengan a un puerto, se redireccionen a la IP fija de la Raspberry. Ahora todas las peticiones que se envíen a la dirección IP de mi casa,

Conexión del sensor de ritmo cardiaco

Para la conexión del sensor, primero tenemos que entender que función tiene cada uno de los pines, para posteriormente saber dónde hay que conectarlo.

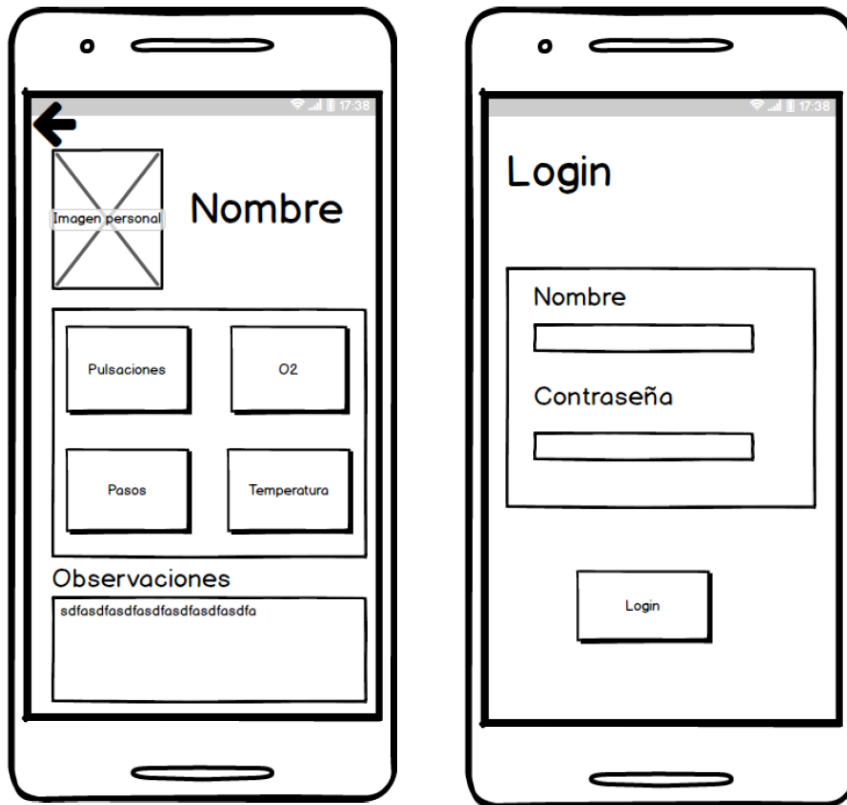


Software

Creación de los mockups

43

pantallas e incluso pude añadir algunos detalles, como, por ejemplo, que al pulsar en cierto botón fueras a una pantalla determinada, esto dota a los mockups de un dinamismo con el que puedes ver si la navegación de la aplicación es fluida. Aquí adjunto algunos ejemplos de las pantallas:

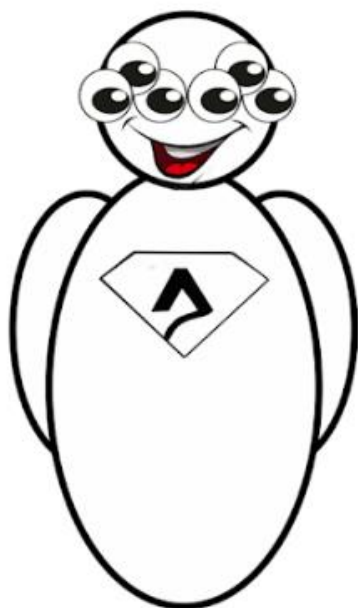


Diseño de Argos

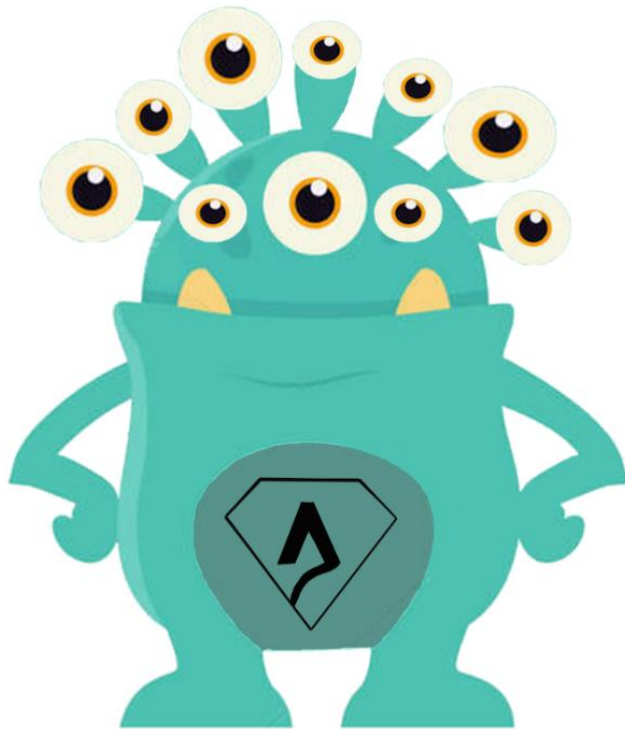
En cuanto a Argos, no se podía quedar sólo con un nombre, necesitaba darle una cara. Es por esto, que comencé a diseñar una imagen de Argos, pero adaptada a los nuevos tiempos, algo minimalista, sencillo y fácil de introducir en la app. En un principio Argos era algo así:



Como se puede observar, era un diseño quizás demasiado simple, por eso, decidí darle una vuelta más, añadiéndole un logo, que más tarde podría usar como icono de la app. Además, quería darle un toque de caricatura, un poco más como los dibujos animados. Por ello, diseñe la siguiente versión de Argos.

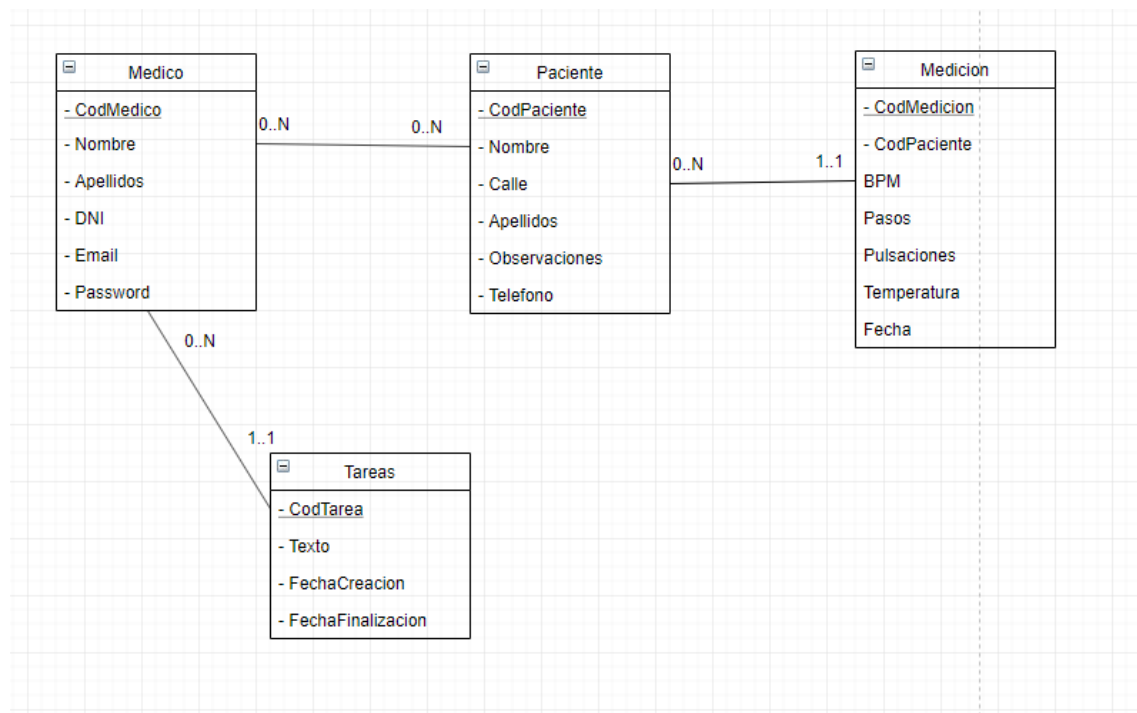


Finalmente, después de las recomendaciones que una compañera me dio, decidí que era necesario darle una vuelta más al diseño y ponerle color, además de que esos colores, marcarían los colores de la aplicación. Por ello, tenía que escogerlos de una manera muy cuidadosa. En cuanto al diseño, continúe optando por acercarlo más a los dibujos animados, y después de muchas pruebas y errores, aquí está la versión final de Argos:




Para la elección de los colores he seguido la norma del 60,30,10 la cual también he llevado a la práctica en la aplicación móvil. Esta norma la suelen usar algunos diseñadores, para escoger, estructurar y darle proporción a los colores escogidos. Básicamente, lo que nos dice esta regla es que tienes que escoger un color dominante y usarlo en el 60% del espacio, otro secundario que ocupe un 30% y finalmente un último color para el 10% restante.

Diseño de BD



Aquí tenemos el diseño de la pequeña base de datos de la que se ha hecho uso para el proyecto. Como podemos observar un médico, tiene diversos atributos necesarios como el nombre, apellidos, su código, etc. Además, un médico puede no tener tareas, pero en caso de tenerlas puede tener muchas. Una tarea si existe, tiene que estar asociada a uno y sólo un médico. Un médico puede o no tener pacientes, en caso de tener, puede tener muchos. Un paciente, puede o no tener un médico, en caso de tener, puede tener muchos médicos. Un paciente, puede o no tener mediciones, en caso de tener puede tener varias. Una medición tiene que tener asociado un paciente, en el caso de que exista.

Diseño de la API

medicos Todo lo referido a la BD de los medicos		▼
POST	/login	Login para los medicos 
POST	/registro	Registro para medicos
POST	/tareas	Trae el listado de tareas que tiene el medico
pacientes Acceso a la administración de los pacientes		▼
GET	/pacientes	Trae el listado de pacientes del medico que ha hecho login
GET	/pacientes/{id}	Trae los detalles del paciente
DELETE	/pacientes/{id}	Elimina un paciente de la lista
mediciones Mediciones de los sensores		▼
POST	/medicion/{idUserario}	Guarda la medición del sensor de un paciente
GET	/medicion/{idUserario}	Guarda la medición del sensor de un paciente

Aquí se presentan los endpoints que se han dispuesto en la API para la realización del proyecto. Como podemos ver, tenemos los típicos endpoints de login y registro, necesarios para el cliente móvil. En estos endpoints, lo que hacemos es enviar en el body de la petición http los datos necesarios, una vez se han comprobado los datos, se devuelve un token, generado con [JWT](#)(JSON Web Token). Este token contiene datos del usuario, y es necesario para acceder a los demás endpoints, ya que, se ha implementado un middleware, que comprueba que las peticiones que se hacen (a excepción de login y registro) contengan el token y que este token sea de un usuario que se encuentra en la BD.

Por lo que respecta a los endpoints de los pacientes, estos son consumidos por el cliente móvil y básicamente lo que se realiza es una petición a la API con los datos del médico que ha hecho login o se ha registrado. En el primer caso `‘/pacientes’`, una vez se ha comprobado que los datos del médico son correctos, se busca cuáles son los pacientes que tiene asociados, y se le envía a la aplicación móvil en formato JSON. En caso de que queramos más detalles de un paciente, tenemos el endpoint `‘/pacientes/{id}’`, para esta petición es necesario que se le envíe, además de los datos del médico, el id del paciente del cual buscamos su información. Una vez, se ha comprobado que los datos sean correctos, se devolverá los datos del paciente, así como las mediciones de las constantes vitales que se han realizado.

Por último, nos quedan las mediciones, estos endpoints, están hechos para que sean usados por la placa servidor LoRa, la cual es la encargada de enviar a la API, las mediciones que recibe la placa cliente. El primer endpoint `‘/mediciones/{idUsuario}’` lo que hace es registrar las pulsaciones de un usuario en BD, además, en caso de que esas pulsaciones sean anómalas, se envía un aviso al cliente móvil.

BUSINES Y FUTURO

Lo que se busca con este proyecto es hacer ver el potencial que tiene esta nueva tecnología de comunicación y como funciona en un pequeño proyecto, como ha sido este. La idea es que en un futuro alguna empresa relacionada con la salud se interese por el proyecto, e invierta capital en este, con ese capital, se podrían desarrollar mejores prototipos que hagan las mediciones. Por otro lado, a los usuarios se les podría cobrar una tasa mensual, o en caso de que fuera una entidad pública, hacer paquetes de usuarios, es decir, fijar un servicio para una cantidad de usuarios.

A nivel personal, seguiré desarrollando este proyecto poco a poco, ya que me parece que tiene potencial, y que se puede mejorar, tanto a nivel de software como a nivel de hardware. A nivel de software, se puede hacer un cliente web que consuma también la API. También se puede mejorar la aplicación móvil, para que sea más intuitiva y fácil de usar. A nivel de hardware, lo que intentaría sería reducir el tamaño de los sensores, hasta que cupieran en una pulsera.

CONCLUSIONES

Una vez finalizada esta ardua travesía, he llegado a diferentes conclusiones. La primera es lo que cuesta enfrentarse a la programación del hardware, sobre todo si es la primera vez que lo hago. La segunda es que, como ingenieros informáticos, debemos ampliar nuestros horizontes, en mi caso, he acabado la rama de ingeniería software, pero no por ello no voy a quedarme ahí, mi idea es seguir experimentando. Pienso que el saber no ocupa lugar y que hoy en día tenemos al alcance de la mano unas herramientas que hace 40 años la gente no podía imaginar. Por ello, está en nuestra mano el hacer buen uso de ellas, siendo 'buen uso' algo muy ambiguo, ya que lo que es bueno para unos, no lo es para otros. Lo que yo entiendo como un buen uso de estas herramientas, es hacer un uso que tenga cierta repercusión social, a modo de facilitar la vida a la gente en la medida de lo posible.

REFERENCIAS Y BIBLIOGRAFIA

- Luces inteligentes: <https://www.efectoled.com/blog/la-iluminacion-inteligente/>
- Asistentes virtuales: https://es.wikipedia.org/wiki/Asistente_virtual
- Asistentes virtuales: <http://tublogtecnologico.com/los-asistentes-virtuales-dispositivos-pueden-hacerte-la-vida-mas-comoda/>
- Siri: <https://es.wikipedia.org/wiki/Siri>
- Google Now: https://es.wikipedia.org/wiki/Google_Now
- Alexa: <https://tecnologiaencasa.com/amazon-alexa/>
- Arduino: <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>
- Git: <https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git>
- Trello: <http://gestron.es/que-es-trello/>
- Kanban: <http://www.pmoinformatica.com/2014/03/sistema-kanban-desarrollo-de-software.html>
- VSCode: <https://blogs.itpro.es/eduardocloud/2016/08/22/visual-studio-code-que-es-y-que-no-es/>
- VSCode: https://es.wikipedia.org/wiki/Visual_Studio_Code
- React: <https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html>
- React-Native: https://en.wikipedia.org/wiki/React_Native
- React-Native: <https://clouddistrict.com/blog-dev/que-es-react-native/>
- NodeJS: <https://www.netconsulting.es/blog/nodejs/>
- MySQL: <https://es.wikipedia.org/wiki/MySQL>
- CSS: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- Copcar: <http://www.copcar.es/Medicina-preventiva.html>
- Arduino IDE: <https://tuelectronica.es/que-es-arduino-ide/>
- Holter: <https://www.webconsultas.com/pruebas-medicas/holter-12058>
- Lucia: <https://www.esmartcity.es/comunicaciones/comunicacion-sistema-inteligente-alerta-monitorizacion-personas-mayores-dependientes-viven-solas-proyecto-lucia>
- Photoshop: <https://neoattack.com/neowiki/photoshop/>
- MongoDB: <https://www.mongodb.com/>
- Imágenes Google Maps: <https://www.calcmaps.com/es/map-radius/>
- Raspberry Pi: https://www.elconfidencial.com/tecnologia/2013-11-22/dos-millones-de-razones-para-saber-que-es-exactamente-raspberry-pi_56003/
- LoRa: <https://www.semtech.com/lora/what-is-lora>
- LoRa WAN: <https://lorawan.es/>
- Lora Sandeep Mistry: <https://github.com/sandeepmistry/arduino-LoRa>
- Pines placa LoRa: <https://github.com/Xinyuan-LilyGO/TTGO-LoRa-Series?spm=a2g0o.detail.1000023.17.69c06566r5zpWk>
- FreeMyIP: <https://freemyip.com/>

- Raspberry Pi: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>
- Heroku: <https://devcenter.heroku.com/articles/getting-started-with-nodejs#set-up>
- Argos: https://es.wikipedia.org/wiki/Argos_Panoptes
- Consejos en el diseño: <https://pickaso.com/2017/consejos-color-tipografia-apps>
- Ley del 1%: <https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy/1300>
- LoRa transceivers: <https://www.semtech.com/products/wireless-rf/lora-transceivers>
- Semtech SX1261: <https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1261>
- Semtech LLCC68: <https://www.semtech.com/products/wireless-rf/lora-transceivers/llcc68>
- Proyecto SIDRA: http://www.sstalcahuano.cl/sidra/somos_sidra.php
- Repositorio API: <https://github.com/JaimeSarrion/ArgosAPI>
- Repositorio app: <https://github.com/JaimeSarrion/ArgosApp>
- Repositorio hardware: <https://github.com/JaimeSarrion/ArgosHardware>